PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-297433
(43)Date of publication of application : 11.10.2002

(51)Int.Cl. G06F 12/00
G06F 17/30

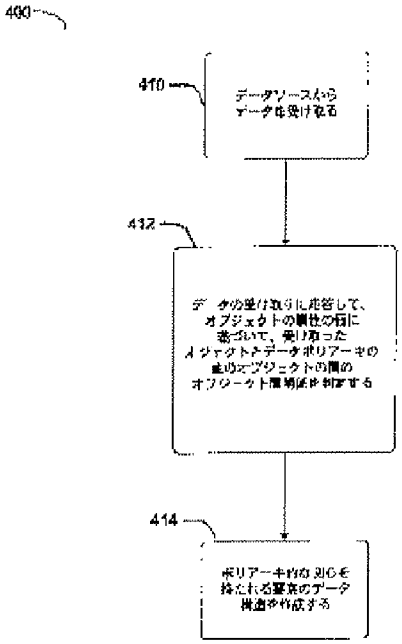
(21)Application number : 2001-365074
(22)Date of filing : 29.11.2001

(71)Applicant : MICROSOFT CORP
(72)Inventor : CAMERON KIM
ROBERTSON GEORGE G
BROWN MARK R

(30)Priority
Priority number : 2000 250344 Priority date : 30.11.2000 Priority country : US

(54) DYNAMIC GENERATING MULTIPLE HIERARCHIES OF INTER-OBJECT RELATIONSHIPS BASED ON OBJECT ATTRIBUTE VALUES

(57)Abstract:
PROBLEM TO BE SOLVED: To provide dynamic generating multiple hierarchies of inter-object relationships based on object attribute values.
SOLUTION: The described arrangements and procedures dynamically generate a data polyarchy from information received from a data store (e.g. a directory or database). The data polyarchy represents multiple hierarchies of inter- object relationships based on values of attributes of the objects. These multiple hierarchies are generated and represented in a manner that is independent of object naming and predetermined static hierarchical data structures.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－297433
（Ｐ2002－297433Ａ）
(43)公開日　平成14年10月11日(2002.10.11)

(51)Int.Cl.[7]	識別記号	ＦＩ	テーマコード(参考)
Ｇ０６Ｆ　12/00	５４５	Ｇ０６Ｆ　12/00	５４５Ａ　５Ｂ０７５
17/30	１１０	17/30	１１０Ｃ　５Ｂ０８２
	４１９		４１９Ａ

審査請求　未請求　請求項の数83　ＯＬ　外国語出願　（全124頁）

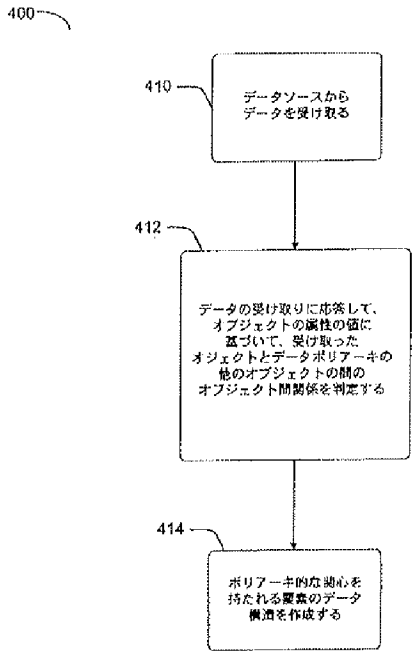
(21)出願番号　特願2001－365074(Ｐ2001－365074)
(22)出願日　平成13年11月29日(2001.11.29)
(31)優先権主張番号　６０／２５０，３４４
(32)優先日　平成12年11月30日(2000.11.30)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　391055933
マイクロソフト　コーポレイション
ＭＩＣＲＯＳＯＦＴ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国　ワシントン州　98052－6399　レッドモンド　ワン　マイクロソフト　ウェイ　（番地なし）
(74)代理人　100077481
弁理士　谷　義一　（外２名）

最終頁に続く

(54)【発明の名称】　オブジェクト属性値に基づくオブジェクト間関係の複数階層の動的生成

(57)【要約】
【課題】　オブジェクト属性値に基づくオブジェクト間関係の複数階層の動的生成を提供する。
【解決手段】　説明される構成および手順によって、データストア（たとえばディレクトリまたはデータベース）から受け取る情報からデータポリアーキが動的に生成される。データポリアーキは、オブジェクトの属性の値に基づくオブジェクト間関係の複数階層を表す。これらの複数階層は、オブジェクト命名および所定の階層データ構造から独立の形で生成され、表現される。



1

【特許請求の範囲】

【請求項１】　分散コンピューティング環境における方法であって、

データストアから、複数のオブジェクトに対応するデータを受け取るステップと、

前記データの受取に応答して、前記オブジェクトの属性の値に基づきオブジェクト間関係の複数の階層を動的に生成するステップであって、前記オブジェクト間関係の複数の階層がデータポリアーキであるステップとを備えることを特徴とする方法。

【請求項２】　請求項１に記載の方法であって、前記データストアが、ディレクトリまたはデータベースを備えることを特徴とする方法。

【請求項３】　請求項１に記載の方法であって、前記データポリアーキが、オブジェクト間関係の交差する階層を備えることを特徴とする方法。

【請求項４】　請求項１に記載の方法であって、前記データポリアーキが、弾力性のあるオブジェクト間関係を備えることを特徴とする方法。

【請求項５】　請求項１に記載の方法であって、前記オブジェクト間関係の複数の階層を動的に生成するステップが、

前記オブジェクト中の第１オブジェクトと第２オブジェクトとの間の１つ以上の次元関係の中の１つの次元関係を識別するステップと、

前記１つの次元関係に関して、前記第１オブジェクトが前記第２オブジェクト内で表現されるように、前記第１オブジェクトを前記第２オブジェクトに挿入するステップとをさらに備えることを特徴とする方法。

【請求項６】　請求項１に記載の方法であって、前記オブジェクト中の第１オブジェクトおよび第２オブジェクトが、前記データポリアーキ内で別々のエンティティとしてそれぞれ表現され、前記オブジェクト間関係の複数の階層を動的に生成するステップが、

前記第１オブジェクトと前記第２オブジェクトとの間の１つ以上の次元関係の中の１つの次元関係を識別するステップと、

前記１つの次元関係に関して、前記第１オブジェクトへのリンクを前記第２オブジェクトに挿入するステップとをさらに備えることを特徴とする方法。

【請求項７】　請求項６に記載の方法であって、前記リンクが、ジャンプゲートであることを特徴とする方法。

【請求項８】　請求項１に記載の方法であって、前記オブジェクト間関係の複数の階層が、オブジェクト命名から独立に、かつ所定の階層データ構造から独立に表現されることを特徴とする方法。

【請求項９】　請求項１に記載の方法であって、前記オブジェクト間関係が、単一方向オブジェクト関係および両方向オブジェクト関係を表すことを特徴とする方法。

【請求項１０】　請求項１に記載の方法であって、前記

2

データポリアーキが、多対多オブジェクト関係の参照解除された次元ナビゲーションを提供するメンバシップ階層を備えることを特徴とする方法。

【請求項１１】　請求項１に記載の方法であって、前記データポリアーキを生成するステップが、

前記オブジェクト中の第１、第２および第３オブジェクトの間の多対多オブジェクト関係の参照解除された次元ナビゲーションを容易にするために、前記第１および第２オブジェクトを前記第３オブジェクトに関係させるステップをさらに備えることを特徴とする方法。

【請求項１２】　請求項１に記載の方法であって、自然言語を用いて、前記データポリアーキ内のオブジェクト間関係に命名するステップをさらに備えることを特徴とする方法。

【請求項１３】　請求項１に記載の方法であって、前記データポリアーキを生成するステップが、前記オブジェクトの個々の１つについて、該オブジェクトの個々の１つにアクセスする方法を示すために、複数の述部を確立するステップをさらに備えることを特徴とする方法。

【請求項１４】　請求項１に記載の方法であって、前記データポリアーキを生成するステップが、前記オブジェクトの個々の１つについて、該オブジェクトの個々の１つをインデクシングするために、複数のドメインプロパティを確立するステップをさらに備えることを特徴とする方法。

【請求項１５】　請求項１４に記載の方法であって、前記ドメインプロパティが、データタイプ、データ精度表示、スケール表示、およびヌル可能性表示を備えることを特徴とする方法。

【請求項１６】　請求項１に記載の方法であって、前記データポリアーキを生成するステップが、属性を備えるオブジェクトに関する提示または検索のためのストラテジを確立するために、前記オブジェクトの属性の相対分布を判定するステップをさらに備えることを特徴とする方法。

【請求項１７】　請求項１に記載の方法であって、前記データポリアーキを生成するステップが、

属性を備えるオブジェクトに関する提示または検索のためのストラテジを確立するために、前記オブジェクトの属性の相対分布を判定するステップをさらに備え、前記ストラテジが、

前記オブジェクト中のデフォルト検索オブジェクトを見つける第１動作、

前記オブジェクト中の特定のオブジェクトを突き止める第２動作、

前記オブジェクト中の特定のオブジェクトに対応するデータ関係のデフォルト階層を得る第３動作、

前記オブジェクト中の特定のオブジェクトに対応するデータ関係の特定の階層を得る第４動作、

前記オブジェクト中の特定のオブジェクトに対応するデ

ータ関係の複数の階層の少なくとも１つのサブセットを識別する第５動作、および前記オブジェクト中の特定のオブジェクトに対応するデータ関係の複数の階層を得る第６動作のうちの１つ以上の動作を備えることを特徴とする方法。

【請求項１８】　請求項１７に記載の方法であって、前記ストラテジが、再帰アクセスストラテジまたは線形スキャンアクセスストラテジを備えることを特徴とする方法。

【請求項１９】　請求項１７に記載の方法であって、ドメインプロパティが論理ドメインプロパティを備え、該論理ドメインプロパティが、特徴的ドメイン、位置決めドメイン、または分類ドメインを備えることを特徴とする方法。

【請求項２０】　請求項１に記載の方法であって、各オブジェクトが１つ以上の各々の属性をさらに備え、前記データポリアーキを生成するステップが、

複数の特徴的属性を識別するステップであって、各特徴的属性が、階層のルートである前記オブジェクト中の各オブジェクトを表し、各特徴的属性が、前記オブジェクトにまたがる類似する属性の実質的に一意の分布から得られるものであるステップと、

前記オブジェクト中のあるオブジェクトの検索を狭めるために１つ以上の位置決め属性を識別するステップであって、各位置決め属性が、前記オブジェクトにまたがる類似する属性の相対的に大きい分布から得られるものであるステップと、

あるオブジェクトの検索からオブジェクトをフィルタアウトするために１つ以上の分類属性を識別するステップであって、各分類属性が、前記オブジェクトにまたがる類似する属性の相対的に小さい分布から得られるものであるステップとをさらに備えることを特徴とする方法。

【請求項２１】　ディレクトリベースのオブジェクトのオブジェクト間関係を表すコンピュータであって、

プロセッサと、

前記プロセッサに結合されたメモリとを備え、該メモリが、コンピュータ実行可能命令およびデータを備え、前記プロセッサが、前記コンピュータ実行可能命令を取り出して実行し、前記コンピュータ実行可能命令が、

データストアから、複数のオブジェクトに対応するデータを受け取る命令と、

前記データの受取に応答して、オブジェクトの属性の値に基づきオブジェクト間関係の複数の階層を動的に生成する命令であって、前記オブジェクト間関係の複数の階層がデータポリアーキである命令とを備えることを特徴とするコンピュータ。

【請求項２２】　請求項２１に記載のコンピュータであって、前記データストアが、ディレクトリまたはデータベースを備えることを特徴とするコンピュータ。

【請求項２３】　請求項２１に記載のコンピュータであって、前記データポリアーキが、オブジェクト間関係の交差する階層を備えることを特徴とするコンピュータ。

【請求項２４】　請求項２１に記載のコンピュータであって、前記データポリアーキが、弾力性のあるオブジェクト間関係を備えることを特徴とするコンピュータ。

【請求項２５】　請求項２１に記載のコンピュータであって、前記オブジェクト間関係の複数の階層を動的に生成するコンピュータ実行可能命令が、

前記オブジェクト中の第１オブジェクトと第２オブジェクトとの間の１つ以上の次元関係の中の１つの次元関係を識別する命令と、

前記１つの次元関係に関して、前記第１オブジェクトが前記第２オブジェクト内で表現されるように、前記第１オブジェクトを前記第２オブジェクトに挿入する命令とをさらに備えることを特徴とするコンピュータ。

【請求項２６】　請求項２１に記載のコンピュータであって、前記オブジェクト中の第１オブジェクトおよび第２オブジェクトが、前記データポリアーキ内で別々のエンティティとしてそれぞれ表現され、前記オブジェクト間関係の複数の階層を動的に生成するコンピュータ実行可能命令が、

前記第１オブジェクトと前記第２オブジェクトとの間の１つ以上の次元関係の中の１つの次元関係を識別する命令と、

前記１つの次元関係に関して、前記第１オブジェクトへのリンクを前記第２オブジェクトに挿入する命令とをさらに備えることを特徴とするコンピュータ。

【請求項２７】　請求項２６に記載のコンピュータであって、前記リンクが、ジャンプゲートであることを特徴とするコンピュータ。

【請求項２８】　請求項２１に記載のコンピュータであって、前記オブジェクト間関係の複数の階層が、オブジェクト命名から独立に、かつ所定の階層データ構造から独立に表現されることを特徴とするコンピュータ。

【請求項２９】　請求項２１に記載のコンピュータであって、前記オブジェクト間関係が、単一方向オブジェクト関係および両方向オブジェクト関係を表すことを特徴とするコンピュータ。

【請求項３０】　請求項２１に記載のコンピュータであって、前記データポリアーキが、多対多オブジェクト関係の参照解除された次元ナビゲーションを提供するメンバシップ階層を備えることを特徴とするコンピュータ。

【請求項３１】　請求項２１に記載のコンピュータであって、前記データポリアーキを生成するコンピュータ実行可能命令が、

前記オブジェクト中の第１、第２および第３オブジェクトの間の多対多オブジェクト関係の参照解除された次元ナビゲーションを容易にするために、前記第１および第２オブジェクトを前記第３オブジェクトに関係させる命令をさらに備えることを特徴とするコンピュータ。

5

【請求項３２】　請求項２１に記載のコンピュータであって、前記データポリアーキを生成するコンピュータ実行可能命令が、前記オブジェクトの個々の１つについて、該オブジェクトの個々の１つにアクセスする方法を示すために、複数の述部を確立する命令をさらに備えることを特徴とするコンピュータ。

【請求項３３】　請求項２１に記載のコンピュータであって、前記データポリアーキを生成するコンピュータ実行可能命令が、前記オブジェクトの個々の１つについて、該オブジェクトの個々の１つをインデクシングするために、複数のドメインプロパティを確立する命令をさらに備えることを特徴とするコンピュータ。

【請求項３４】　請求項３３に記載のコンピュータであって、前記ドメインプロパティが、データタイプ、データ精度表示、スケール表示、およびヌル可能性表示を備えることを特徴とするコンピュータ。

【請求項３５】　請求項２１に記載のコンピュータであって、前記データポリアーキを生成するコンピュータ実行可能命令が、属性を備えるオブジェクトに関する提示または検索のためのストラテジを確立するために、前記オブジェクトの属性の相対分布を判定する命令をさらに備えることを特徴とするコンピュータ。

【請求項３６】　請求項２１に記載のコンピュータであって、前記データポリアーキを生成するコンピュータ実行可能命令が、
属性を備えるオブジェクトに関する提示または検索のためのストラテジを確立するために、前記オブジェクトの属性の相対分布を判定する命令をさらに備え、前記ストラテジが、
前記オブジェクト中のデフォルト検索オブジェクトを見つける第１動作、
前記オブジェクト中の特定のオブジェクトを突き止める第２動作、
前記オブジェクト中の特定のオブジェクトに対応するデータ関係のデフォルト階層を得る第３動作、
前記オブジェクト中の特定のオブジェクトに対応するデータ関係の特定の階層を得る第４動作、
前記オブジェクト中の特定のオブジェクトに対応するデータ関係の複数の階層の少なくとも１つのサブセットを識別する第５動作、および前記オブジェクト中の特定のオブジェクトに対応するデータ関係の複数の階層を得る第６動作のうちの１つ以上の動作を備えることを特徴とするコンピュータ。

【請求項３７】　請求項３６に記載のコンピュータであって、前記ストラテジが、再帰アクセスストラテジまたは線形スキャンアクセスストラテジを備えることを特徴とするコンピュータ。

【請求項３８】　請求項３６に記載のコンピュータであって、ドメインプロパティが論理ドメインプロパティを備え、該論理ドメインプロパティが、特徴的ドメイン、

6

位置決めドメイン、または分類ドメインを備えることを特徴とするコンピュータ。

【請求項３９】　請求項２１に記載のコンピュータであって、各オブジェクトが１つ以上の各々の属性をさらに備え、前記データポリアーキを生成するコンピュータ実行可能命令が、
複数の特徴的属性を識別する命令であって、各特徴的属性が、階層のルートである前記オブジェクト中の各オブジェクトを表し、各特徴的属性が、前記オブジェクトにまたがる類似する属性の実質的に一意の分布から得られるものである命令と、
前記オブジェクト中のあるオブジェクトの検索を狭めるために１つ以上の位置決め属性を識別する命令であって、各位置決め属性が、前記オブジェクトにまたがる類似する属性の相対的に大きい分布から得られるものである命令と、
あるオブジェクトの検索からオブジェクトをフィルタアウトするために１つ以上の分類属性を識別する命令であって、各分類属性が、前記オブジェクトにまたがる類似する属性の相対的に小さい分布から得られるものである命令とをさらに備えることを特徴とするコンピュータ。

【請求項４０】　データ構造であって、
複数の仮想オブジェクトデータフィールドであって、各仮想オブジェクトデータフィールドが、データストア内の複数のオブジェクト中の各オブジェクトに対応し、前記仮想オブジェクトデータフィールドが、前記オブジェクトの属性に基づくオブジェクト間関係の複数の階層を示す仮想オブジェクトデータフィールドを備えることを特徴とするデータ構造。

【請求項４１】　請求項４０に記載のデータ構造であって、前記データストアが、ディレクトリまたはデータベースであることを特徴とするデータ構造。

【請求項４２】　請求項４０に記載のデータ構造であって、各仮想オブジェクトデータフィールドが、
前記データストア内の対応するオブジェクトを一意に識別するための第１のグローバル一意識別子（ＧＵＩＤ）データフィールドをさらに備えることを特徴とするデータ構造。

【請求項４３】　請求項４０に記載のデータ構造であって、仮想オブジェクトデータフィールドの１つが、前記オブジェクト中の第１オブジェクトに対応し、前記仮想オブジェクトデータフィールドの１つが、前記第１オブジェクトのサブ要素として前記オブジェクト中の第２オブジェクトを一意に識別するためのエンティティ参照データフィールドをさらに備え、該エンティティ参照データフィールドが、前記データストア内で前記第２オブジェクトを一意に識別することを特徴とするデータ構造。

【請求項４４】　請求項４３に記載のデータ構造であって、前記エンティティ参照が、ＧＵＩＤであることを特徴とするデータ構造。

【請求項４５】　請求項４０に記載のデータ構造であって、各仮想データオブジェクトデータフィールドが、１つ以上の述部データフィールドをさらに備え、各述部データフィールドが、オブジェクト間関係の１つ以上の階層に関して特定のオブジェクトを提示するための各動作を示すことを特徴とするデータ構造。

【請求項４６】　請求項４０に記載のデータ構造であって、各仮想データオブジェクトデータフィールドが、オブジェクト間関係の１つ以上の階層に関して前記オブジェクト中の対応するオブジェクトをインデクシングするためのドメインプロパティデータフィールドをさらに備えることを特徴とするデータ構造。

【請求項４７】　請求項４６に記載のデータ構造であって、前記ドメインプロパティデータフィールドが、データタイプ、データ精度表示、スケール表示、またはヌル可能性表示を備える物理ドメインと、一意ドメイン、位置決めドメイン、または分類ドメインを備える論理ドメインとをさらに備えることを特徴とするデータ構造。

【請求項４８】　請求項４０に記載のデータ構造を格納したことを特徴とするコンピュータ可読媒体。

【請求項４９】　コンピュータ可読媒体であって、データストアから、複数のオブジェクトに対応するデータを受け取るコンピュータ実行可能命令と、前記データの受取に応答して、前記オブジェクトの属性の値に基づきオブジェクト間関係の複数の階層を動的に生成するコンピュータ実行可能命令であって、前記オブジェクト間関係の複数の階層がデータポリアーキであるコンピュータ実行可能命令とを備えることを特徴とするコンピュータ可読媒体。

【請求項５０】　請求項４９に記載のコンピュータ可読媒体であって、前記データストアが、ディレクトリまたはデータベースを備えることを特徴とするコンピュータ可読媒体。

【請求項５１】　請求項４９に記載のコンピュータ可読媒体であって、前記データポリアーキが、オブジェクト間関係の交差する階層を備えることを特徴とするコンピュータ可読媒体。

【請求項５２】　請求項４９に記載のコンピュータ可読媒体であって、前記データポリアーキが、弾力性のあるオブジェクト間関係を備えることを特徴とするコンピュータ可読媒体。

【請求項５３】　請求項４９に記載のコンピュータ可読媒体であって、前記データポリアーキが、該データポリアーキ内の１つ以上のサブオブジェクトに関係する複合オブジェクトを備え、オブジェクト間関係を判定するコンピュータ実行可能命令が、前記複合オブジェクトを独立サーフェスエンティティとして表現する命令と、前記独立サーフェスエンティティ内の１つ以上のサブオブジェクトを別々のエンティティとして参照する命令であって、前記１つ以上のサブオブジェクトが、オブジェクト命名から独立に、かつ前記サーフェスエンティティと前記１つ以上のサブオブジェクトとの間の階層データ関係から独立に参照される命令とをさらに備えることを特徴とするコンピュータ可読媒体。

【請求項５４】　請求項４９に記載のコンピュータ可読媒体であって、前記データポリアーキが、該データポリアーキ内の１つ以上のサブオブジェクトに関係する第１オブジェクトを含み、オブジェクト間関係を判定するコンピュータ実行可能命令が、前記第１オブジェクトをサーフェスエンティティとして表現する命令と、前記１つ以上のサブオブジェクトのそれぞれを、前記サーフェスエンティティから独立した別々の各エンティティとして表現する命令と、前記１つ以上のサブオブジェクトのそれぞれにおいて、いかなるオブジェクト命名または階層関係から独立に、前記サーフェスエンティティを参照する命令とをさらに備えることを特徴とするコンピュータ可読媒体。

【請求項５５】　請求項４９に記載のコンピュータ可読媒体であって、前記オブジェクト間関係の複数の階層が、オブジェクト命名から独立に、かつ所定の階層データ構造から独立に表現されることを特徴とするコンピュータ可読媒体。

【請求項５６】　請求項４９に記載のコンピュータ可読媒体であって、前記オブジェクト間関係が、単一方向オブジェクト関係および両方向オブジェクト関係を表すことを特徴とするコンピュータ可読媒体。

【請求項５７】　請求項４９に記載のコンピュータ可読媒体であって、前記データポリアーキが、多対多オブジェクト関係の参照解除された次元ナビゲーションを提供するメンバシップ階層を備えることを特徴とするコンピュータ可読媒体。

【請求項５８】　請求項４９に記載のコンピュータ可読媒体であって、前記データポリアーキを生成するコンピュータ実行可能命令が、前記オブジェクト中の第１、第２および第３オブジェクトの間の多対多オブジェクト関係の参照解除された次元ナビゲーションを容易にするために、前記第１および第２オブジェクトを前記第３オブジェクトに関係させる命令をさらに備えることを特徴とするコンピュータ可読媒体。

【請求項５９】　請求項４９に記載のコンピュータ可読媒体であって、前記データポリアーキを生成するコンピュータ実行可能命令が、前記オブジェクトの個々の１つについて、該オブジェクトの個々の１つにアクセスする方法を示すために、複数の述部を確立する命令をさらに備えることを特徴とするコンピュータ可読媒体。

【請求項６０】　請求項４９に記載のコンピュータ可読

9

媒体であって、前記データポリアーキを生成するコンピュータ実行可能命令が、属性を備えるオブジェクトに関する提示または検索のためのストラテジを確立するために、前記オブジェクトの属性の相対分布を判定する命令をさらに備えることを特徴とするコンピュータ可読媒体。

【請求項６１】　請求項４９に記載のコンピュータ可読媒体であって、各オブジェクトが、１つ以上の各々の属性をさらに備え、前記データポリアーキを生成するコンピュータ実行可能命令が、

複数の特徴的属性を識別する命令であって、各特徴的属性が、階層のルートである前記オブジェクト中の各オブジェクトを表し、各特徴的属性が、前記オブジェクトにまたがる類似する属性の実質的に一意の分布から得られるものである命令と、

前記オブジェクト中のあるオブジェクトの検索を狭めるために１つ以上の位置決め属性を識別する命令であって、各位置決め属性が、前記オブジェクトにまたがる類似する属性の相対的に大きい分布から得られるものである命令と、

あるオブジェクトの検索からオブジェクトをフィルタアウトするために１つ以上の分類属性を識別する命令であって、各分類属性が、前記オブジェクトにまたがる類似する属性の相対的に小さい分布から得られるものである命令とをさらに備えることを特徴とするコンピュータ可読媒体。

【請求項６２】　請求項４９に記載のコンピュータ可読媒体であって、前記データポリアーキを生成するコンピュータ実行可能命令が、前記オブジェクトの個々の１つについて、該オブジェクトの個々の１つをインデクシングするために、複数のドメインプロパティを確立する命令をさらに備えることを特徴とするコンピュータ可読媒体。

【請求項６３】　請求項６２に記載のコンピュータ可読媒体であって、前記ドメインプロパティが、データタイプ、データ精度表示、スケール表示、およびヌル可能性表示を備えることを特徴とするコンピュータ可読媒体。

【請求項６４】　請求項４９に記載のコンピュータ可読媒体であって、前記データポリアーキを生成するコンピュータ実行可能命令が、

属性を備えるオブジェクトに関する提示または検索のためのストラテジを確立するために、前記オブジェクトの属性の相対分布を判定する命令をさらに備え、前記ストラテジが、

前記オブジェクト中のデフォルト検索オブジェクトを見つける第１動作、

前記オブジェクト中の特定のオブジェクトを突き止める第２動作、

前記オブジェクト中の特定のオブジェクトに対応するデータ関係のデフォルト階層を得る第３動作、

10

前記オブジェクト中の特定のオブジェクトに対応するデータ関係の特定の階層を得る第４動作、

前記オブジェクト中の特定のオブジェクトに対応するデータ関係の複数の階層の少なくとも１つのサブセットを識別する第５動作、および前記オブジェクト中の特定のオブジェクトに対応するデータ関係の複数の階層を得る第６動作のうちの１つ以上の動作を備えることを特徴とするコンピュータ可読媒体。

【請求項６５】　請求項６４に記載のコンピュータ可読媒体であって、前記ストラテジが、再帰アクセスストラテジまたは線形スキャンアクセスストラテジを備えることを特徴とするコンピュータ可読媒体。

【請求項６６】　請求項６４に記載のコンピュータ可読媒体であって、ドメインプロパティが論理ドメインプロパティを備え、該論理ドメインプロパティが、特徴的ドメイン、位置決めドメイン、または分類ドメインを備えることを特徴とするコンピュータ可読媒体。

【請求項６７】　ディレクトリベースのオブジェクトのオブジェクト間関係を表すコンピュータであって、

データストアから、複数のオブジェクトに対応するデータを受け取る手段と、

前記データの受取に応答して、前記オブジェクトの属性の値に基づきオブジェクト間関係の複数の階層を動的に生成する手段であって、前記オブジェクト間関係の複数の階層がデータポリアーキである手段とを備えることを特徴とするコンピュータ。

【請求項６８】　請求項６７に記載のコンピュータであって、前記データポリアーキが、オブジェクト間関係の交差する階層を備えることを特徴とするコンピュータ。

【請求項６９】　請求項６７に記載のコンピュータであって、前記データポリアーキが、弾力性のあるオブジェクト間関係を備えることを特徴とするコンピュータ。

【請求項７０】　請求項６７に記載のコンピュータであって、前記オブジェクト間関係の複数の階層を動的に生成する手段が、

前記オブジェクト中の第１オブジェクトと第２オブジェクトとの間の１つ以上の次元関係の中の１つの次元関係を識別する手段と、

前記１つの次元関係に関して、前記第１オブジェクトが前記第２オブジェクト内で表現されるように、前記第１オブジェクトを前記第２オブジェクトに挿入する手段とをさらに備えることを特徴とするコンピュータ。

【請求項７１】　請求項６７に記載のコンピュータであって、前記オブジェクト中の第１オブジェクトおよび第２オブジェクトが、前記データポリアーキ内で別々のエンティティとしてそれぞれ表現され、前記オブジェクト間関係の複数の階層を動的に生成する手段が、

前記第１オブジェクトと前記第２オブジェクトとの間の１つ以上の次元関係の中の１つの次元関係を識別する手段と、

前記１つの次元関係に関して、前記第１オブジェクトへのリンクを前記第２オブジェクトに挿入する手段とをさらに備えることを特徴とするコンピュータ。

【請求項７２】　請求項７１に記載のコンピュータであって、前記リンクが、ジャンプゲートであることを特徴とするコンピュータ。

【請求項７３】　請求項６７に記載のコンピュータであって、前記オブジェクト間関係の複数の階層が、オブジェクト命名から独立に、かつ所定の階層データ構造から独立に表現されることを特徴とするコンピュータ。

【請求項７４】　請求項６７に記載のコンピュータであって、前記データポリアーキが、多対多オブジェクト関係の参照解除された次元ナビゲーションを提供するメンバシップ階層を備えることを特徴とするコンピュータ。

【請求項７５】　請求項６７に記載のコンピュータであって、前記データポリアーキを生成する手段が、

前記オブジェクト中の第１、第２および第３オブジェクトの間の多対多オブジェクト関係の参照解除された次元ナビゲーションを容易にするために、前記第１および第２オブジェクトを前記第３オブジェクトに関係させる手段をさらに備えることを特徴とするコンピュータ。

【請求項７６】　請求項６７に記載のコンピュータであって、前記データポリアーキを生成する手段が、前記オブジェクトの個々の１つについて、該オブジェクトの個々の１つにアクセスする方法を示すために、複数の述部を確立する手段をさらに備えることを特徴とするコンピュータ。

【請求項７７】　請求項６７に記載のコンピュータであって、前記データポリアーキを生成する手段が、前記オブジェクトの個々の１つについて、該オブジェクトの個々の１つをインデクシングするために、複数のドメインプロパティを確立する手段をさらに備えることを特徴とするコンピュータ。

【請求項７８】　請求項７７に記載のコンピュータであって、前記ドメインプロパティが、データタイプ、データ精度表示、スケール表示、およびヌル可能性表示を備えることを特徴とするコンピュータ。

【請求項７９】　請求項６７に記載のコンピュータであって、前記データポリアーキを生成する手段が、属性を備えるオブジェクトに関する提示または検索のためのストラテジを確立するために、前記オブジェクトの属性の相対分布を判定する手段をさらに備えることを特徴とするコンピュータ。

【請求項８０】　請求項６７に記載のコンピュータであって、前記データポリアーキを生成する手段が、

属性を備えるオブジェクトに関する提示または検索のためのストラテジを確立するために、前記オブジェクトの属性に上ってとられる値の相対分布を判定する手段をさらに備え、前記ストラテジが、

前記オブジェクト中のデフォルト検索オブジェクトを見つける第１動作、

前記オブジェクト中の特定のオブジェクトを突き止める第２動作、

前記オブジェクト中の特定のオブジェクトに対応するデータ関係のデフォルト階層を得る第３動作、

前記オブジェクト中の特定のオブジェクトに対応するデータ関係の特定の階層を得る第４動作、

前記オブジェクト中の特定のオブジェクトに対応するデータ関係の複数の階層の少なくとも１つのサブセットを識別する第５動作、および前記オブジェクト中の特定のオブジェクトに対応するデータ関係の複数の階層を得る第６動作のうちの１つ以上の動作を備えることを特徴とするコンピュータ。

【請求項８１】　請求項８０に記載のコンピュータであって、前記ストラテジが、再帰アクセスストラテジまたは線形スキャンアクセスストラテジを備えることを特徴とするコンピュータ。

【請求項８２】　請求項８０に記載のコンピュータであって、ドメインプロパティが論理ドメインプロパティを備え、該論理ドメインプロパティが、特徴的ドメイン、位置決めドメイン、または分類ドメインを備えることを特徴とするコンピュータ。

【請求項８３】　請求項６７に記載のコンピュータであって、各オブジェクトが１つ以上の各々の属性をさらに備え、前記データポリアーキを生成する手段が、

複数の特徴的属性を識別する手段であって、各特徴的属性が、階層のルートである前記オブジェクト中の各オブジェクトを表し、各特徴的属性が、前記オブジェクトにまたがる類似する属性の実質的に一意の分布から得られるものである手段と、

前記オブジェクト中のあるオブジェクトの検索を狭めるために１つ以上の位置決め属性を識別する手段であって、各位置決め属性が、前記オブジェクトにまたがる類似する属性の相対的に大きい分布から得られるものである手段と、

あるオブジェクトの検索からオブジェクトをフィルタアウトするために１つ以上の分類属性を識別する手段であって、各分類属性が、前記オブジェクトにまたがる類似する属性の相対的に小さい分布から得られるものである手段とをさらに備えることを特徴とするコンピュータ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本願は、参照によって本明細書に組み込まれる、２０００年１１月３０日出願の米国仮特許出願第６０／２５０３４４号の恩恵を主張する。

【０００２】説明される内容は、オブジェクト間関係に関する。具体的には、内容は、オブジェクトの属性の値に基づくオブジェクト間関係の複数階層の動的生成に関する。

【０００３】

13

【従来の技術】オブジェクトは、明示的または暗示的なオブジェクト間関係を形成するために、異なるオブジェクトに関してリンクするか、相関させるか、関連付けるか、区別するか、何らかの形でカテゴリ化することができる。たとえば、組織では、人が、通常は、組織内の他人、組織リソース（たとえばプリンタ、ファクシミリなど）、地理的位置、業務編成単位、クラブメンバシップなどとの明示的または暗示的な関係を有する。各々のオブジェクト（すなわち人、他人、リソースなど）間の明示的または暗示的な関係のそれぞれが、各々の階層データ関係を表す。

【０００４】たとえば、１つの階層データ関係は、特定のリソース（たとえば会社構内の建物、部屋、プリンタなど）にアクセスできる、会社内の人のそれぞれによって表され、リソースが階層のルートノードであり、リソースへのアクセスを有する個人が葉である。もう１つの階層データ関係は、会社の管理構造を構成する個人によって表される。他のオブジェクト間データ関係は、特定の業務編成単位内の個人の階層、特殊なトレーニングを受けた会社内の全従業員などを表すことができる。

【０００５】

【発明が解決しようとする課題】残念ながら、他のデータの階層内のオブジェクト間関係を表すために、ネットワーク管理者によってある程度までデータストアを構成することができるが、複雑なオブジェクト間関係（たとえば、複数の階層内の単一のオブジェクトを表すものなど）は、従来のデータストア（たとえばディレクトリ、データベースなど）のシステムおよびテクノロジを使用して簡単かつ適切には表現されない。従来のディレクトリには、周知のＸ．５００標準規格およびライトウェイト・ディレクトリ・アクセス・プロトコル（ＬＤＡＰ：Ｌｉｇｈｔｗｅｉｇｈｔ　Ｄｉｒｅｃｔｏｒｙ　Ａｃｃｅｓｓ　Ｐｒｏｔｏｃｏｌ）が含まれる。

【０００６】従来のデータストアシステムおよびデータストアテクノロジのこの限界を示すために、ディレクトリが、通常は、硬直したデータ命名スキーマおよび柔軟でないディレクトリスキーマを使用してオブジェクト間関係を表すことを検討されたい。ディレクトリ内のオブジェクトまたはノードは、ルートノードを階層の最上部とする単一の階層内で編成される。ルートノードは、名前を有する。ディレクトリ内の他のノードのそれぞれは、ルートノードに対する直接の命名関係に基づいて、各々のノードの階層内に介在するノードのそれぞれに関して、命名される。その結果、親オブジェクトが単一の動作で名前を変更される場合に、その親オブジェクトに従属するか子であるすべてのオブジェクトも、同一の単一の動作で名前を変更される。これは、オブジェクトの完全な「識別名」に、ルートノードの名前までのすべての各親オブジェクトの名が含まれるからである。

【０００７】オブジェクトの静的な位置またはデータストア内の他の各オブジェクトに関するデータ関係をも表すのが、オブジェクトの完全な識別名である。したがって、オブジェクトの識別名は、単一の階層内のオブジェクト命名をその階層内のオブジェクト間関係と、柔軟でない形でからみ合わせる。このために、データストアのナビゲーションは、オブジェクト間関係を判定し、後に提示するために、最上部から最下部へ、すなわち、ルートオブジェクトから、親オブジェクトへ、さらにすべての従属する子オブジェクトへ、実行しなければならない。

14

【０００８】従来のデータストア（たとえばディレクトリ、データベースなど）は、注意深く指定された柔軟でないオブジェクト命名方式に頼って、オブジェクト間関係を識別するので、データストアを構成する管理者は、データストアを構成する時に、オブジェクト間関係の先見的知識を必要とする。さらに、データストアの構成では、データストア内のオブジェクト間関係の正しい表現だけではなく、データストアをナビゲートするために検索エンジンが必要とするヒューリスティックも考慮しなければならない。

【０００９】さらに困ったことに、弾力性のある（ｅｌａｓｔｉｃ）データ関係は、従来のデータストアのシステムおよび技法を使用しては、簡単に記述、表現、またはナビゲートされない。弾力性のあるデータ関係とは、関係が、任意の時点のオブジェクトを定義するデータから導出されるデータ関係である。これは、経時的に、弾力性のあるデータ関係が動的になる可能性があることを意味する。たとえば、次の、明白でなく、潜在的に弾力性のある、データ関係を検討されたい。ウェブサイトとそのウェブサイトを構成するウェブページ、顧客とその顧客が商人から購入する個性を与えられたサービス、パーソナルコンピュータ（ＰＣ）とＰＣに結合された周辺装置、都市と都市内の行政区、会社と会社の連絡先、従業員と従業員に従属する者、などである。

【００１０】これらの明白でなく、潜在的に弾力性のあるデータ関係は、簡単には表現されない。というのは、サーフェスオブジェクトと対応するサブオブジェクトとの間の１対１対応をデータストア内で表現する必要がある時に、必ず不可逆的な設計選択を行わなければならないからである（従来の慣例は、ディレクトリ方式変更の深刻な性質に起因して、ディレクトリ方式更新を厳密に制御することである）。ネットワーク管理者は、サブオブジェクトをディレクトリスキーマ内でサーフェスオブジェクトの属性として表現することによって、特定のオブジェクトへのサブオブジェクトの「完全な組込み」を選ぶことができる。もしくは、ネットワーク管理者は、サブオブジェクトコンポーネントについてスキーマ内で別々のオブジェクトを作成し、別々のオブジェクトをサーフェスオブジェクトの下位に位置決めすることによって、各オブジェクトの「完全な区別」を選ぶことができ

る。

【００１１】この不可逆的な設計選択を例示するために、特定のネットワークルータに、ルータのバックプレーンに挿入された複数のルータモジュールが含まれると想定されたい。ルータおよびルータモジュールに関する情報は、通常は、１つまたは２つの異なる形（それぞれが、エンティティおよびその各々の他のエンティティに対する関係を表現する方法に同等に不満足に依存する可能性がある）でディレクトリに保管される。設計選択の１つが、ルータおよびその対応するルータモジュールの特徴を、親オブジェクトとしてネットワークルータを表す単一の階層データ構造として表し、対応するルータモジュールの特徴を、親オブジェクトに従属する子オブジェクトとして表すことである。異なる設計選択は、ルータおよびルータの関連するルータモジュールの特徴を、複合属性を有する単一の親オブジェクトとして表すことである。この親オブジェクトは、ルータ（バックプレーン）を表し、複合属性が、ルータによってホストされる各々のルータモジュールを表す。

【００１２】第１の設計選択を考慮すると、ルータおよびモジュールが構成される方法によっては、バックプレーン上のルータモジュールまたは基板に関する情報の縮小が、非実際的であることがわかる場合がある。これは、ルータのバックプレーンの機能性が、ルータによってホストされるネットワークルータモジュールの機能性と比較して小さい可能性があるからである。それに対して、第２の設計選択を考慮すると、バックプレーンから基板を完全に分離することが、同様に不満足である可能性がある。というのは、ルータが、一般に複数のルータモジュールを含む単一の物理的ルータボックスであるからである。

【００１３】柔軟性のないディレクトリスキーマを用いてデータ関係を表現するための上で説明した解決策の両方が、実施に時間がかかり、反直観的である。最も単純な設計問題を解決する前に、ディレクトリの形状および命名のセマンティクスに合意しなければならない。したがって、エンティティと、対応するサブエンティティとの間の１対１対応を従来のディレクトリで表現する必要がある時には、必ず、不可逆的で柔軟でない設計選択を、ディレクトリスキーマ内で行わなければならない。

【００１４】どの設計選択を選択する場合であっても、データストアおよび、オブジェクト間関係に関してデータストア内のオブジェクトをナビゲートし、検索し、表現するのに使用されるツールは、かなり制限されてきた。これは、データストア自体が、オブジェクトの可能な明示的および暗示的なオブジェクト間関係のすべてを表現できないからである。これは、多くのコンピュータプログラマによって、従来のディレクトリでのディレクトリスキーマの最も手におえない問題の１つと見なされる。これは、コンピュータプログラムアプリケーションが、通常は、ディレクトリプラットフォームまたはディレクトリインスタンスにまたがってポータブルでない理由とも見なされる。

【００１５】さらに困ったことに、情報テクノロジでの最近の開発によって、ネットワーク管理者に、データの異種のデータソース（たとえばデータベース、ディレクトリなど）を単一の論理ディレクトリまたは「メタディレクトリ」に結び合わせる機会が与えられる。そのような異種のデータベースおよびディレクトリには、たとえば、企業ユーザ、有形および無形のリソース、財務情報、会社の電子メールシステム、ネットワークオペレーティングシステムなどに対応する情報が含まれる。

【００１６】メタディレクトリによって、ネットワーク管理者に、データストアを構成し管理する従来のシステムおよび手順を使用して簡単にまたは適切に記述、表現、ナビゲート、または提示することができない、複雑でしばしば弾力性のあるオブジェクトデータ関係が提示される。データストアを構成するために、管理者（または管理者の一員）の側でかなりの労力が要求される。そのようなオブジェクト間関係（動的であるか否かを問わず）の手動の判定および実装は、人的なエラーおよび見落としの潜在性をはらんでいる。さらに、そのようなディレクトリ構成を実行するのに適当なレベルのそのような知識を有するデータベース管理者は、費用がかかる。

【００１７】以下で説明する内容は、オブジェクト間関係の表現のこれらおよび他の問題に対処するものである。

【００１８】

【課題を解決するための手段】本明細書で説明する構成および手順によって、データストア（たとえばディレクトリまたはデータベース）から受け取る情報からデータポリアーキ（ｄａｔａｐｏｌｙａｒｃｈｙ）が動的に生成される。データポリアーキは、オブジェクトの属性の値に基づくオブジェクト間関係の複数階層を表す。これらの複数階層は、オブジェクト命名および所定の静的階層データ構造とは独立の形で生成され、表現される。

【００１９】

【発明の実施の形態】（概要）以下の内容は、ディレクトリオブジェクト命名およびオブジェクト間関係が、からみ合わされ、複雑なデータ関係を表現するには非実際的である、単一の静的階層内での複雑な実世界のオブジェクトの提示という従来の観念に代わるものである。具体的に言うと、静的オブジェクト間関係の単一のディレクトリ内でオブジェクト間関係を表現するための識別名という従来の観念が、オブジェクトの属性に基づく複数次元のデータ関係（たとえば単一方向または両方向もしくはその両方の関係）での弾力性のある（非静的）オブジェクト間接続のグラフに置換される。言い換えると、データ関係によって、１つ以上のデータオブジェクトが、オブジェクト間関係の１つ以上の各々の次元または

17

ポリアーキに参加することが確立される。１つ以上のこれらの階層が、交差することができ、オブジェクト間関係の交差する階層が作成される。

【００２０】オブジェクト属性に基づいて動的に生成されるデータ関係の複数階層は、データポリアーキ内で表現される。具体的に言うと、データポリアーキは、各オブジェクトの各々のデータ属性またはデータ値と、ポリアーキ的データセット内の他のオブジェクトに対応する属性とのこれらの値のさまざまな相互関係とを使用して生成される。データポリアーキ内のオブジェクト間関係は、オブジェクト間関係が任意の時点のオブジェクトによって定義されるデータから導出されるので、弾力性を有することができる。オブジェクト間の関係のパターンは、データ関係の１つ以上の「次元」またはポリアーキ内でオブジェクトを提示することによって明らかになる。そのような関係は、オブジェクトを表現する仮想エンティティ間のオブジェクト間接続を使用して提示される。仮想エンティティは、関心を持たれるオブジェクトに対応し、関心を持たれるオブジェクトに関する情報をさらに得る方法に関する情報を含む、関心を持たれるオブジェクトに関する情報を含み、これを編成する。そのようなオブジェクトを、その関心を具体化する人またはコンピュータプログラムに提示することができる。

【００２１】データストア内のオブジェクト間関係を表現する従来のシステムおよび手順と異なって、下で説明する構成および手順は、自動化されており、オブジェクト間関係を構成するためにネットワーク管理者からの手動の介入を必要としないという点で、動的である。データポリアーキを動的に生成することによって、オブジェクトデータに基づく複雑なオブジェクト間関係が、スキーマ設計者に柔軟でない設計選択を提示せずに自動的に判定される。

【００２２】これは、ネットワーク管理者またはコンピュータプログラム（たとえば検索エンジン）が、データストアの生成、ナビゲーション、または検索のために、複雑なオブジェクト間関係の先見的知識を有する必要がないことを意味する。これは、データストア内の各オブジェクトを、各々のオブジェクトに適用されるものと同数の、多数の異なる次元を有するオブジェクト間関係から見ることができることも意味する。さらに、オブジェクトの弾力性のあるデータ関係が変化する際に、データポリアーキによって、これらの変更が自動的に検出され、反映される。

【００２３】以下の説明では、請求項に記載の要素を組み込んだオブジェクト間関係のポリアーキを表現するディレクトリ方式に基づく構成および手順を示す。その内容は、法定要件を満足するための限定性を有する形で説明される。しかし、説明自体は、本特許の範囲を制限することを意図されていない。そうではなくて、本発明は、他の現在または将来のテクノロジと共に、本明細書

18

に記載の要素と異なる要素または類似する要素の組み合わせを含めるために、請求される内容を他の形でも実施できることが企図されている。

【００２４】（例示的システム）図１に、オブジェクトの属性の値に基づいてオブジェクト間関係の複数階層を動的に生成し管理する例示的システム１００を示す。このシステムは、１つ以上の他の任意選択のデータサーバ１０６、１つ以上のデータベース１０８、および１つ以上のクライアントコンピュータ１１０にネットワーク１０４を介して機能的に結合されるデータポリアーキサーバ１０２を含む分散コンピューティング環境を表す。ネットワークへのデータポリアーキサーバの機能的結合は、１つ以上のサーバ設備（たとえば、ウェブファームーサーバファームの外部のサーバ設備）、企業ポータル（イントラネット）、ローカルエリアネットワーク（ＬＡＮ）、同一位置に配置されたデータストア（たとえばデータベース１０８）などを介して複数の異なる形で行うことができる。

【００２５】データポリアーキサーバ１０２には、コンピュータ実行可能命令１１６およびデータ１１８を含むメモリ１１４に機能的に結合されたプロセッサ１１２が含まれる。プロセッサは、コンピュータ実行可能命令を取り出し、実行し、そのような実行中にデータを取り出すように構成される。そのようなコンピュータ実行可能命令には、オペレーティングシステム（図示せず）と、オブジェクトの属性の値に基づいてオブジェクト間関係の複数階層を動的に生成し、管理するためのデータポリアーキ管理モジュール１２０が含まれる。これらの動的に生成されるオブジェクト間関係の複数階層は、データポリアーキとも称するポリアーキ的データセット１２２に保管される。データポリアーキ１１８を生成するために、データポリアーキ管理モジュール１２０は、１つ以上の他の任意選択のサーバ１０６またはデータベース１０８からなどの任意の数の異なるデータソースからのデータ（たとえばエクステンシブル・マークアップ言語（ＸＭＬ：Ｅｘｔｅｎｓｉｂｌｅ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ）データ）を使用する。たとえば、サーバ１０６が、任意の数のさまざまなデータストア（データベース、ディレクトリ、メタディレクトリなど）からデータポリアーキサーバにデータ（たとえば企業ユーザのディレクトリ、リソースのディレクトリ、財政情報、企業電子メールシステムのディレクトリ、ネットワークオペレーティングシステムのディレクトリなど）を供給する。データベース１０８は、構造化データストアまたは非構造化データストアであり、これには、ＸＭＬデータベース、ハイパーテキストマークアップ言語（ＨＴＭＬ）データベース、ＳＱＬサーバデータベース、などのオブジェクト指向データベースが含まれる。

【００２６】データポリアーキ１２２の生成および管理に応答して、管理モジュール１２０は、それぞれ、関心

19

を持たれる要素のスキーマ（ｅｌｅｍｅｎｔｓ　ｏｆ　ｉｎｔｅｒｅｓｔ　ｓｃｈｅｍａ）１２４の要素の生成および更新を行う。関心を持たれる要素のスキーマによって、任意選択のクライアントコンピュータ１１０が、オブジェクト間関係の各々のポリアーキに関してデータポリアーキ内のオブジェクトを操作でき、表示できる方法が示される。

【００２７】たとえば、関心を持たれる要素のスキーマ１２４によって、データポリアーキ１２２内の各オブジェクトが、各々のオブジェクトを表現する仮想エンティティ（たとえば図２の仮想オブジェクト２１０を参照されたい）を参照するアドレスとして識別される。これらの仮想エンティティは、アドレスのベクトルまたは配列としてスキーマに保管される。データポリアーキ内のオブジェクトが有することのできる属性の異なるタイプのそれぞれも、さまざまな属性タイプに対してどの種類のインデックスが使用されるかと同様に、スキーマで識別される（データインデックスによって、オブジェクトアクセスが提供される）。属性タイプのそれぞれについて、その定義と共に、それに対応するインデックスを保管することが便利である。この形で、たとえば、誰かが属性について要求する場合に、インデックスが簡単に使用可能になり、属性がとる値のすべてを、非常にすばやく判定することができる（関心を持たれる要素のスキーマは、下で、図３に関してさらに詳細に説明する）。

【００２８】データポリアーキサーバ１０２は、任意の数のスキーマ１２４を生成することができる。生成されたスキーマのそれぞれは、他のスキーマ１２４によって表現されるオブジェクトから独立に、データポリアーキ１２２内のオブジェクトのさまざまなサブセットへのアクセスを提供することができる。たとえば、第１のスキーマ１２４をネットワーク管理者に配布して、それ以外の場合に他の従業員から保護されるか隠される、プリンタおよびアクセスリストなどのリソースおよび属性へのアクセスを提供することができる。同一の形で、人的資源について、第２のスキーマを社長に配布することができる。第２のスキーマは、社長に、ある特権従業員レコードへのアクセスを与えることができるが、第２のスキーマは、第１のスキーマを介してネットワーク管理者グループが使用可能なリソースに関して、完全に不活動にすることができる。この形で、スキーマ１２４を、組織のリソースへのアクセス制御を提供するように設計することができる。

【００２９】データポリアーキサーバ１０２は、関心を持たれる要素のスキーマ１２４を１つ以上の任意選択のクライアント１１０に通信する。クライアントコンピュータは、関心を持たれる要素のスキーマによる記述に従ってデータポリアーキ１２２内のオブジェクト間関係を表示するグラフィカルユーザインターフェース（図示せず）をサポートする。データ関係のポリアーキ内のオブジェクトを表示する例示的な構成および手順は、本願の譲受人に譲渡され、参照によって組み込まれる、関連米国特許出願第０９／７２８９３５号明細書、表題「階層ポリアーキ可視化（Ｈｉｅｒａｒｃｈｙ　Ｐｏｌｙａｒｃｈｙ　Ｖｉｓｕａｌｉｚａｔｉｏｎ）」、２０００年１１月２９日出願に記載されている。

20

【００３０】データポリアーキ１２２および関心を持たれる要素のスキーマ１２４は、データポリアーキサーバ１０２によって、メモリキャッシュ１１４に１回または複数回複製することができる。例示的なメモリキャッシュを、下で図１４に関して詳細に説明する。ポリアーキ的サーバは、対応するメモリキャッシュからのデータセットのいずれかを操作することができるので、各々のデータセットの必要な個数のコピーを設けることができる。したがって、クライアント１１０がどれほど要求していても、データポリアーキサーバはその要求を満足することができる。

【００３１】データポリアーキ１２２および関心を持たれる要素のスキーマ１２４が、データポリアーキサーバ１０２によってメモリキャッシュ１１４内に複製される時に、サーバは、オブジェクト間関係の最も新しいまたは現行の表現を表す、信頼すべきストア（図示せず）をメモリ１１４内で維持することができる。そのような信頼すべきストアが有益であるのは、キャッシュが、まさにそれらの性質によって、必ずある程度は古く、キャッシュ内のデータが、最も最近のキャッシュ更新と同程度に「新鮮」またはタイムリーであるに過ぎないからである。これに鑑みて、データポリアーキ１２２に情報を要求するクライアントは、クライアントによって要求されるデータ信頼性またはタイムリーさのレベルを示すことができる。高度なタイムリーさが要求される場合には、サーバ１０２は、データキャッシュからではなく、信頼すべきストアからデータポリアーキにアクセスすることができる。信頼すべきキャッシュへのアクセスの速度は、その各々の実施形態（たとえば、サーバの内部でランダムアクセスメモリで実施されるか、サーバの外部でデータ記憶装置内で実施されるか）に依存する。

【００３２】（例示的なデータポリアーキ）図２に、データ内の属性に基づくオブジェクト間関係の複数次元を表す例示的なポリアーキデータセット１２２を示す。データとは、区別することができるものである（たとえば、ディレクトリ、データベースなどで表現された関心を持たれるオブジェクトであるすべてのものをオブジェクトとすることができる）。データセット１２２は、設計者が彼ら自身のカスタマイズされたタグを作成でき、アプリケーション間および組織間のデータの定義、伝送、検証、および解釈を可能にするようにフォーマットされる。たとえば、データフォーマットを、ＸＭＬデータフォーマットとすることができる。

【００３３】データポリアーキ１２２には、複数の仮想

21

オブジェクトデータフィールド２１０が含まれる。各仮想オブジェクトデータフィールドには、たとえば各々のオブジェクトに関する情報をさらに得る方法に関する情報を含む、各々のオブジェクトに関する情報が含まれ、編成される。具体的に言うと、仮想オブジェクトには、グローバル一意識別子（ＧＵＩＤ）データフィールド２１２と、特定のオブジェクトに適当である場合に、１つ以上の属性データフィールド２１４が含まれる。

【００３４】ＧＵＩＤ２１２によって、これまたは他のデータボリアーキ１２２内のこのオブジェクトまたは他のオブジェクトに関して仮想オブジェクト（各々のオブジェクトを表す）が識別される。既に述べたように、これらのオブジェクトは、１つ以上の物理的に分散したデータストア内で表現することができ、これらのデータストアは、１つ以上のディレクトリサービスならびに１つ以上のデータボリアーキによって論理的に集中化される。属性データフィールド２１４によって、仮想オブジェクト２１０のすべてのデータ属性またはデータ値が定義される。各属性は、仮想オブジェクトの実際のインスタンスに含めることができる属性に対応する。そのような属性には、たとえば、１つ以上の述部（ｐｒｅｄｉｃａｔｅ）データフィールド２１６、複数のドメインプロパティデータフィールド２１８、および０個以上のサブオブジェクトエンティティ参照２２０が含まれる。

【００３５】各述部データフィールド２１６は、他のオブジェクトの１つ以上の階層に関する特定のオブジェクトのアクセスまたは提示のための各々の動作を示す（各オブジェクトは、ボリアーキ的データセット１２２内の仮想オブジェクト２１０によって表される）。そのような動作には、１つ以上の別個のタイプの検索（たとえば、線形検索と再帰検索）、データ変換（たとえば、ある階層関係から別の異なる階層関係への）などが含まれる（図１３のブロック１３１８を参照されたい）。

【００３６】オブジェクトが、サブオブジェクトエンティティ２２０を参照しないことを意味する単純オブジェクトである場合には、述部２１６動作（たとえば、検索、修正、仮想オブジェクト２１０からのものなどの構造から他のオブジェクトの階層内のオブジェクトへのデータ変換）が、関心を持たれる各々のオブジェクトに対応する。しかし、オブジェクトが、１つ以上のサブオブジェクトに対するデータ関係を有することを意味する複合オブジェクトである場合には、述部動作が、オブジェクトまたは１つ以上のサブオブジェクトの組み合わせに対応する。

【００３７】ドメインプロパティデータフィールド２１８には、物理ドメインプロパティと論理ドメインプロパティが含まれる。物理ドメインプロパティは、データオブジェクトのインデクシングに使用される１組または複数の組の値を示す。物理ドメインプロパティは、データタイプ、データ精度表示、スケール表示、およびヌル可

22

能性（ｎｕｌｌａｂｉｌｉｔｙ）表示を含む物理ドメインプロパティのグループから選択される。ドメインプロパティ２１８の論理ドメインプロパティ態様は、オブジェクトデータ値を特定のドメインに割り当てられるようにすることによって、データボリアーキ１２２の検索およびナビゲーションを容易にする。具体的に言うと、論理ドメインは、データボリアーキ内の他のオブジェクトに関する対応するオブジェクトのアクセスまたは提示のストラテジを示す。たとえば、論理ドメインプロパティには、一意ドメインプロパティ、位置決めドメインプロパティ、および分類ドメインプロパティが含まれる。ボリアーキ的データ関係管理モジュール１１６がオブジェクトの属性に割り当てる特定の１つの論理ドメインプロパティは、データボリアーキ内の他のオブジェクトの同一の属性の他の値に関する、データボリアーキ内の属性の値の相対分布に基づく。

【００３８】ここで、（ａ）データボリアーキ１２２内の属性がとる値の相対分布と、（ｂ）データ分布によって、どのオブジェクトが各々の次元（階層）、上ノード、および下ノードを表すかが決定される方法とを記述することができる。

【００３９】（相対属性値分布）属性が有する値の組は、その属性の論理ドメインの一部である。データボリアーキ１１１内の値の実際の分布（各潜在的な値を含む潜在的なオブジェクトの数に関する）に関して収集される情報も、属性の論理ドメインのプロパティである。属性値の相対分布を判定するために、１つ以上の閾値（たとえば、下閾値と上閾値）を定義して、ボリアーキ内の他のオブジェクトの他の属性に関するデータボリアーキ１２２内の属性の相対分布を判定する。閾値は、データがその中にある比率の誤差を有する可能性がある（たとえば１％の誤差）という前提に基づく（他の統計分析技法を、閾値と組み合わせてまたはその代わりに使用して、オブジェクト属性分布を判定することができる）。

【００４０】たとえば、オブジェクトが、データボリアーキ１２２にロードされる（たとえば１つ以上のディレクトリサーバまたはデータベースサーバ１０６から）時に、データボリアーキ管理モジュール１２０が、閾値に基づいて各オブジェクトの各々の属性値を検査して、（ａ）どの属性がデータセット内のオブジェクトでの分布に関して実質的に一意であるか、（ｂ）どの属性がオブジェクトの実質的に大きい組にわたって分布するか、および（ｃ）どの属性がデータセット内のオブジェクトの実質的に小さい組にわたって分布するかを判定する。これらの判定は、データがある比率の誤差を有するという前提に基づいて行われる。

【００４１】あるデータ誤差というこの前提を念頭において、実質的に一意の属性が、必ずしもデータボリアーキ１２２内でのこの種類の唯一の属性ではないことを検討されたい。そうではなくて、属性が、絶対的に一意で

23

ある場合があり、また、属性が、データセット内の類似する属性の比較的疎な分布に属する場合がある。データセット内のオブジェクトにまたがる分布に関して実質的に一意と判定される属性は、それが他の属性と比較して特徴的であることを示す一意の論理ドメインプロパティを有する。

【００４２】特徴的である属性によって、ポリアーキ的データセット１２２内の各々の一意の次元を識別することができ、この次元が、相互接続されたグラフの上ノードとして表され、この上ノードが、階層次元を表す。このモデルの内部では、デフォルトポリアーキがフラットである。特徴的でない属性は、データセット内のオブジェクトの実質的に大きい組にまたがって分布するか、その代りに、オブジェクトの実質的に小さい組にまたがって分布する。非特徴的な属性は、次元を定義する属性のよい候補ではない。そうではなくて、そのような分布は、非特徴的な属性が１つ以上の識別された次元に属することを示す。したがって、非特徴的な属性は、属性分布によって識別される少なくとも１つの次元の下ノードとして表される。上ノードポリアーキも、下ノードオブジェクトのすべての値が、実質的に一意の上ノードオブジェクト内で突き止められる時に、発見される。

【００４３】オブジェクトの実質的に大きい組にまたがって分布する属性は、位置決めドメインプロパティを有する（たとえば、姓は、位置決めドメインプロパティになる可能性がある）。位置決めドメインプロパティを有する属性は、データポリアーキ１２２内のデータオブジェクトの特定の１つに関する検索を絞りこむのに使用される。オブジェクトの実質的に小さい組にまたがって分布する属性は、分類ドメインプロパティを有する。分類ドメインプロパティを有する属性は、検索手順またはナビゲーション手順から、望ましくないオブジェクトをフィルタアウトするのに使用される。

【００４４】（ジャンプゲート）ＧＵＩＤなどのサブオブジェクトエンティティ参照２２０は、仮想オブジェクト２１０（すなわち、各々のオブジェクト）が、データポリアーキ１２２内の異なるオブジェクトに対する関係を有するかどうかを示すだけではなく、異なるオブジェクトを参照する（すなわち、仮想オブジェクトに対応する異なるオブジェクトを介して）。具体的に言うと、サブオブジェクト参照によって、異なる関心を持たれるオブジェクトが、仮想オブジェクトデータフィールドのサブオブジェクトとして一意に識別される。このサブオブジェクト参照によって、１つ以上のデータストアにまたがって、異なる関心を持たれるオブジェクトが一意に識別される。

24

【００４５】サブオブジェクトを参照する（対応するサブオブジェクトエンティティ参照２２０を介して）仮想オブジェクト２１０が、「ジャンプゲート」である。ジャンプゲートは、ポリアーキ的データセット１２２内の複合オブジェクトと関連するサブオブジェクトとの間の弾力性のあるデータ関係を表す。データポリアーキ内のオブジェクト間データ関係は、１つ以上の単純オブジェクト２１０または複合オブジェクト２１０を用いてモデル化される。オブジェクトが、１つ以上のサブデータ関係を有する場合には、そのような関係は、オブジェクト（または「サーフェスエンティティ」）内で参照されるサブオブジェクト２２０として、またはある次元で別のオブジェクト２１０にリンクされた別のオブジェクト２１０としてのいずれかで表現される。

【００４６】これを示すために、従業員とその従業員に従属する者が、ディレクトリストア内のオブジェクトとして表される人である場合を検討されたい。データストア管理者は、各人のさまざまな態様に関する微細な粒度の情報を維持することを求める場合がある。サーフェスエンティティ（従業員）内のサブワールド情報（従属する者に関する）を表すために、下の表１に示された表現を使用することができる。

【００４７】

【表１】

「表１　単一のサーフェスエンティティ内の強いサブワールド情報の例」

```
<person type = "employee" GlueID = "13399">
    <name> John Doe </name>
    <age> 31 </age>
    <sex> male </sex>
    <dependents>
        <person type = "spouse">
            <name> Alice Doe </name>
            <age> 31 </age>
            <sex> female </sex>
        </person>
        <person type = "child">
            <name> Sigmund Doe </name>
            <age> 8 </age>
            <sex> male </sex>
        </person>
    </dependents>
    <occupation> forester </occupation>
</person>
```

【００４８】完全に別個のエンティティで従属する者に関するサブワールド情報を表現するために、Ａｌｉｃｅ　Ｄｏｅおよび Ｓｉｇｍｕｎｄ　Ｄｏｅが、たとえば表２に示されているように独自のＧｌｕｅ　ＩＤ（ＧＵＩ＊ＩＤ２１２）を有する別々のエンティティに分割される。

【００４９】

【表２】

「表２　分離されたオブジェクト／エンティティ表現の例」

```
<person type = "spouse" GlueID = "24889">
    <relatedEmployee> 13399 </relatedEmployee>
    <name> Alice Doe </name>
    <age> 31 </age>
    <sex> female </sex>
</person>
<person type = "child" GlueID = "24890">
    <relatedEmployee> 13399 </relatedEmployee>
    <name> Sigmund Doe </name>
    <age> 8 </age>
    <sex> male </sex>
</person>
```

【００５０】「ｐｅｒｓｏｎ（人）」要素が、Ｊｏｈｎの仮想エンティティ内のサブ要素として存在するか、彼ら自体の独立の仮想エンティティ内のルート要素として存在するかにかかわらずに同一であることに留意されたい。これに関して、ＪｏｈｎＤｏｅのエンティティを、表３に示されているように縮小することができる。

【００５１】

【表３】

「表３　単一エンティティ表現の例」

```
<person type = "employee" GlueID = "13399">
    <name> John Doe </name>
    <age> 31 </age>
    <sex> male </sex>
    <occupation> forester </occupation>
</person>
```

【００５２】表２に示されたエンティティは、「ｄｅｐ

ｅｎｄｅｎｔｓ（従属する者）」次元に沿ってＪｏｈｎに従属する者に関係し、「ｒｅｌａｔｅｄＥｍｐｌｏｙｅｅ（関連従業員）」が、「ジャンプゲートを通過する」ためにＧｌｕｅ　ＩＤに結合される。

【００５３】この２つの極端の間で、Ｊｏｈｎのノード＊＊を内部的に表４に示されるように表現することを想像することができる。

【００５４】

【表４】

「表４　１つ以上の他のエンティティを参照するエンティティの例」

```
<person type = “employee” GlueID = “13399”>
    <name> John Doe </name>
    <age> 31 </age>
    <sex> male </sex>
    <dependents>
        <person GlueID = “24889”/>
        <person GlueID = “24890”/>
    </dependents>
    <occupation> forester </occupation>
</person>
```

【００５５】表４のエンティティを、そのままクライアントに返し、クライアントが、関係するＧｌｕｅ　ＩＤを展開することによってこの情報に追加できるようにすることができる。もしくは、図１のデータポリアーキサーバ１０２などのサーバ自体が、Ｇｌｕｅ　ＩＤを参照解除し、下記のアマルガム（下の表５に示す）を返し、※※従来のディレクトリ実施形態で明白なジャンプゲート問題に対するこの解決策の弾力性を実証することができる。

【００５６】

【表５】

「表５　参照解除された識別情報の例」

```
<person type = “employee” GlueID = “13399”>
    <name> John Doe </name>
    <age> 31 </age>
    <sex> male </sex>
    <dependents>
        <person type = “spouse” GlueID = “24889”>
            <name> Alice Doe </name>
            <age> 31 </age>
            <sex> female </sex>
        </person>
        <person type = “child” GlueID = “24890”>
            <name> Sigmund Doe </name>
            <age> 8 </age>
            <sex> male </sex>
        </person>
    </dependents>
    <occupation> archeologist </occupation>
</person>
```

【００５７】言い換えると、仮想オブジェクト２１０は、（ａ）文字列、整数など、他の要素を参照しない単純オブジェクト（しばしば「単純要素」と称する）、または（ｂ）１つ以上の他の単純要素または複合要素を参照する複合オブジェクト（しばしば「複合要素」と称する）のいずれかとしてモデル化することができる。この形で、ポリアーキ的データセット１２２が、複数の異なる関係表現のいずれについてもいつでも定義することが

できる、弾力性のあるオブジェクト間データ関係を提供する。

【００５８】したがって、最も単純な設計問題を解決する前にディレクトリの形状および命名のセマンティクスに合意しなければならない、手におえないスキーマ問題を有する従来の硬直したディレクトリ実施形態と大きく異なって、ジャンプゲートとしてモデル化されるオブジェクト間データ関係に出会った時に、基本的な設計判断は不要である。ポリアーキ的データセット１２２に基づくディレクトリ木の形状および命名は、ポリアーキ的データセットが設計された後であっても、さまざまな弾力性を有するオブジェクト間関係を表すことによって影響されない。さらに、複合オブジェクトに対する更新／修正も、１つ以上の関係するサブオブジェクトに対する対応する更新をもたらす可能性があり、これを、複合オブジェクトを表す特定の次元ではなく１つ以上の異なる次元で表現することができる。

【００５９】（参照解除された動作に関するデータポリアーキスキーマの最適化）複数のオブジェクトを、参照＊解除された次元グループまたは多対多オブジェクトの検索動作およびナビゲーション動作のために第３のオブジェクトに関係させることができる。たとえば、グループへのメンバシップは、メンバとグループの間の関係に関する情報を含むメンバシップエンティティによって表現される。メンバシップエンティティには、グループを識別するｍｅｍｂｅｒＯｆデータフィールドと、グループメンバを識別するｍｅｍｂｅｒＩｓデータフィールドが含まれる。この実施形態では、そのような一意の識別が、各々のＧＵＩＤ２１２を使用することによって達成される。

【００６０】あるエンティティがグループのメンバであるかどうかを判定するためには、ｍｅｍｂｅｒＩｓがエンティティのＧＵＩＤであり、ｍｅｍｂｅｒＯｆがグループのＧＵＩＤである関係エンティティを検索する。メンバシップ次元は、表６に示されているように定義される。

【００６１】

【表６】

「表６　スキーマのメンバシップ次元の例」

```
<dimension dereferenceElement = "memberIs">
    <name> membership </name>
    <displayName lang = "en"> Membership </displayName>
    <upnodeReferenceElement> memberOf </upnodeReferenceElement>
    <upnodeNamingElement> GlueID </upnodeNamingElement>
</dimension>
```

【００６２】この例では、グループのＧＵＩＤ（表６では「ＧｌｕｅＩＤ」と表されている）が実質的に一意なので、このＧＵＩＤによって、グループが上ノードとして識別され、子が、ｍｅｍｂｅｒＯｆ要素にグループのＧＵＩＤをセットされたメンバシップエンティティとして識別される。データセットを介する従来の「ダウン」ナビゲーションでは、メンバシップエンティティが列挙され、これによって、各個々のメンバシップの性質（たとえば特定のメンバシップが満了する時）に関する有用な情報がもたらされる可能性がある。

【００６３】実際のグループメンバに関する情報を得るためにｍｅｍｂｅｒＩｓ関連付けを使用する、「間接」列挙を実行することも可能である。それを行うには、参照解除にｍｅｍｂｅｒＩｓをセットして、メンバシップ次元内でグループに対する「ダウン」列挙を発行する。この場合、メンバシップエンティティのｍｅｍｂｅｒＩｓが、グループに属する実際のエンティティの参照解除に使用される。したがって、あるエンティティが属するすべてのグループをリストする逆次元を構成することは単純である。この場合、メンバシップエンティティをリストするか、それらを参照解除してグループ自体に関する情報を得ることもできる。

【００６４】したがって、関心を持たれる要素のスキーマ１２４でグループの逆ポリアーキまたは他の多対多オブジェクト間関係を表現するのに、特殊なスキーマ設計は不要である。

【００６５】（例示的なデータポリアーキスキーマ）図３に、クライアントがデータポリアーキ１２２を意味のある形で操作できる方法を示す、図１の例示的なデータポリアーキスキーマ１２４のさらなる態様を示す。このデータポリアーキスキーマを、「関心を持たれる要素の」スキーマとも称する。要素は、オブジェクト属性またはデータ値である。関心を持たれる要素のスキーマ１２４には、データポリアーキに対するクライアント１１０照会を制限するために複数のデータフィールド３１０が含まれる。そのような照会は、図１のデータポリアーキサーバ１０２に通信される。具体的に言うと、そのような照会は、処理のためにポリアーキデータ管理モジュール１２０に通信される。照会は、スキーマ１２４によって表されるオブジェクトの少なくとも１つのサブセットに制限される。

【００６６】要素３１０は、オブジェクト自体ではなく、データポリアーキ１２２内での分布に関するオブジェクトデータの相対スコープを示すオブジェクト表現（すなわち、図２の仮想オブジェクト２１０）である。上で注記したように、これらの仮想エンティティは、ア

ドレスのベクトルまたは配列としてスキーマに保管される。

【００６７】データポリアーキ１２２内のオブジェクト２１０が有することができる属性２１４の異なるタイプのそれぞれも、スキーマ内で識別され、それと同時に、さまざまな属性タイプに対してどの種類のインデックスが使用されるかも、スキーマ内で識別される。

【００６８】要素３１０（たとえばインデックスのタイプ）は、データポリアーキ１２２内の属性がとる値の相対分布に基づいて選択される（属性がとる値の相対分布は、上で図２に関して説明した）。要素３１０には、データポリアーキ１２２内のすべてのオブジェクトに対応する論理ドメインプロパティの少なくとも１つのサブセットが含まれる（論理ドメインプロパティは、上で図２に関して説明した）。要素３１０は、実質的に一意または「特徴的な」論理プロパティインデックスタイプ、位置決め論理プロパティインデックスタイプ、または分類論理プロパティインデックスタイプもしくはこれらの組合せを有する属性を表す。したがって、要素３１０には、特徴的要素３１０－１、位置決め要素３１０－２、および分類要素３１０－３が含まれる。

【００６９】特徴的要素３１０－１（すなわち、特徴的インデックスタイプ）は、データポリアーキ１２２内の属性の間の次元関係のよい候補であり、たとえば、次元または階層内の上ノードを表す（たとえばＧＵＩＤ、位置、従業員番号、コストセンタなど）一意のオブジェクト（すなわち、特徴的要素によってインデクシングされる属性を有するオブジェクト）によって表される。位置決めインデックスタイプ３１０－３または選択インデックスタイプは、データポリアーキ内でオブジェクトを突き止めるためのよい候補であり、たとえば、姓、建物名、肩書、部屋番号、および類似物などの属性によって表される。性の表示（たとえば男または女）などの分類インデックスタイプを有する属性は、データセット内のオブジェクトの検索でオブジェクトをフィルタリングするためのよい候補である。というのは、分類オブジェクトが、他のインデックスタイプに対応する属性を有するオブジェクトの相対分布と比較して、データポリアーキ内での数が比較的少ないからである。

【００７０】関心を持たれる要素のスキーマ１２４は、非常にカスタマイズ可能である。たとえば、ネットワーク管理者は、英語、フランス語、中国語などの名前などの自然言語の名前を、関心を持たれる要素のスキーマ１２４内の要素またはオブジェクトに割り当てることができる。さらに、管理者は、ジャンプゲートおよび表１から４に関して詳細に説明したように、リンクされるが別個のエンティティとして保管するためにサブオブジェクトを指定することができる。この形で、図１および２のポリアーキ的データセット１２２内の、そうでなければルートオブジェクトに直接従属するはずのオブジェクトが、スキーマ内で昇格の資格を有するようになる。この機構は、複数次元（ポリアーキ）と共に、弾力性のあるジャンプゲートを作るのに使用される。

【００７１】表７に、ＸＭＬデータフォーマットでの例示的な関心を持たれる要素のスキーマ１２４を示す。要素３１０の、ＸＭＬ表現以外の他のデータフォーマット表現（たとえば、ＸＭＬの特徴の少なくともサブセットまたはそれより多くの特徴を有する、ＸＭＬの拡張版）が企図されている。このスキーマ表現では、箱に囲まれたテキスト（すなわち、箱の中または直線で囲まれたテキスト）およびセミコロン「；」の後に続くテキストが、対応するコメントを表す。一般に、複数行のコメントは、箱の中に配置される。

【００７２】

【表７】

「表7 関心を持たれる要素のスキーマの例」

```
<WellKnownEntities GlueID="d7a5f1a9-6ba9-48a2-a464-660d82c24b5c">
  ;「WellKnownEntities GlueID」タグは一意なスキーマIDである
    <ElementsOfInterest>          ;スキーマの始まり
      <element name="objectType">     ;属性の名前
        <displayName lang="en" value="Object Type"/>
```

「表示名言語(displayname lang)」タグは、英語(「en」)などの言語(「lang」)およびその言語内の対応する属性値を示す(例えば、cn の英語の表示名(display name)は「Name」である)。スキーマは、単一の属性に対してさえ多くの異なる表示名を示すように、ネットワーク管理者によって構成できることが理解できる(例えば、英語表示名、フランス語表示名、中国語表示名など)。

```
    </element>
      <element name="GlueID"indexType="Distinguishing">
```

「グルーID(GlueID)」要素は、「特徴的(distinguishing)」属性として識別されることに注意されたい。他のインデックス・タイプ(indexTypes)には、位置決め(locating)または分類(classifying)インデックス・タイプが含まれる。各インデックス・タイプは、データ・ポリアーキ内の属性相対分布に基づく。

```
        <displayName lang="en" value="GlueID"/>
    </element>
      <element name ="cn">
        <displayName lang="en" value="Name"/>
    </element>
      <element name="telephoneNumber">
        <displayName lang="en" value="Phone Number"/>
    </element>
      <element name="roomNumber">
        <displayName lang="en" value="Room Number"/>
    </element>
      <element name="uid">
        <displayName lang="en" value="E-mail Alias"/>
    </element>
      <element name="description">
        <displayName lang>="en" value="Description"/>
```

【００７３】　　【表８】

35 36

```
</element>
                <element name="sn"indexType="selecting"
        <displayName lang="en" value="Surname"/>
</element>
    <element      name="givenName"      indexType="locating"
startingSize="20000">
        <displayName lang="en" value="Given Name"/>
</element>
    <element      name="mail"      indexType="distinguishing"
indexStartingSize="20000"indexGrowBy="20000">
        <displayName lang="en" value="E-mail Address"/>
</element>
    <element name="buildingName"indexType="classifying">
      <displayName lang="en" value="Building Name"/>
</element>
    <element name="title"indexType="classifying">
      <displayName lang="en" value="Title"/>
</element>
    <element name="location"indexType="distinguishing">
      <displayName lang="en" value="Location"/>
</element>
    <element name="locationUpnode"/>
    <element name="uniqueIdentifier"indexType="distinguishing">
      <displayName lang="en" value="Employee Number"/>
</element>
    <element name="manager"/>
      <displayName lang="en" value="Manager"/>
</element>
    <element name="costCenter"indexType="distinguishing">
      <displayName lang="en" value="Cost Center"/>
</element>
    <element name="costCenterUpnode"/>
    </ElementsOfInterest>
    <Dimensions>
```

【００７４】

【表９】

37 38

「次元（Dimensions）」タグは、データ・ポリアーキ１２２内で、データ関係の階層におけるルート・ノードを表すオブジェクトを識別するスキーマの一部である。

<dimension> ；次元を示す

<name>costCenter</name> ；次元の名前

<upnodeReferenceElement>costCenterUpnode</upnodeReferenceElement>

「アップノード参照要素（upnodeReferenceElement）」タグは、親の次元の命名要素に存在する値を含む要素を表す。

<dimensionNamingElement>costCenter</dimensionNaming Element>

「次元命名要素（dimensionNamingElement）」タグは、当該次元においてオブジェクトに命名する。

<view>

「ビュー（view）」タグは、当該次元より上または下のオブジェクトを示すことができることを示すことができる（例えば、兄弟（siblings））。

<displayName lang="en">Business Units</displayName>

<SearchType>nodeQuery</SearchType>

「サーチ・タイプ（SearchType）」タグは、クライアントがどのように照会を生成すべきかを示す。もしなければ、クライアントは、単純なリストを返す照会を生成するであろう。もし「ノード照会（nodeQuery）」であれば、生成された照会は、指定された次元における階層の結果を要求するであろう。もし「ノード制約照会（nodeConstraintQuery）」であれば、生成された照会は、指定された次元における指定されたエンティティより下のエンティティのみに制約された階層を要求するであろう。もし「ノード排他照会（nodeExclusiveQuery）」であれば、生成された照会は、指定された次元における第１の指定されたエンティティより下であって、第２かつ後続の指定されたエンティティより下でないものに制約された階層を要求するであろう。

<up>*</up>

「アップ（up）」タグは、当該次元において表示される階層レベル（階層における祖先）の数を示す。この場合、ワイルド・カード「＊」は、この次元におけるすべてのレベルを見ることができることを示す。

【００７５】 【表１０】

39 40

<ElementsList>

これらの属性は、このレベルで表示されるものを示す。このリストは、カスタマイズ可能である。

<element>cn</element>
<element>uid</element>
<element>telephoneNumber</element>
<element>title</element>
<element>buildingName</element>
<element>roomNumber</element>
<element>description</element>
<element>companyCode</element>
<element>costCenter</element>
</ElementsList> </view>
</dimension>
<dimension>
<name>Management</name>
<upnodeReferenceElement>manager</upnodeReferenceElement>
<dimensionNamingElement>uniqueIdentifier</dimensionNamingEleme
nt>
<view>
<displayNamelang="en"Management</displayName>
<displayName lang="fr">Gestion</displayName>
<SearchType>nodeQuery</SearchType>
<up>*</up>
<ElementsList>
<element>cn</element>
<element>uid</element>
<element>telephoneNumber</element>
<element>title</element>
<element>buildingName</element>
<element>roomNumber</element>
</ElementsList>
<selected>true</selected>

【００７６】 【表１１】

41 42

「選択された(selected)」タグは、クライアントに対して、このビューがクライアント・インターフェースにおいてデフォルト(選択された)ビューであることを示す。 </view>

```
<view>
    <displayName                lang="en">Direct
Reports</displayName>
    <SearchType>nodeQuery</SearchType>
    <up>0</up>
```

「<up>0</up>」タグは、クライアントに対して、指定された次元において、指定されたエンティティのいかなる階層祖先も返されるべきではないことを示す次元方向インジケータである。

```
    <down>1</down>
```

「<down>1</down>」タグは、クライアントに対して、指定された次元において、指定されたエンティティのすぐ下の子孫(階層の1レベル)のみが返されるべきであることを示す次元方向インジケータである。

```
    <ElementsList>
        <element>cn</element>
        <element>uid</element>
        <element>telephoneNumber</element>
        <element>title</element>
        <element>buildingName</element>
        <element>roomNumber</element>
    </ElementsList>
</view>
<view>
    <displayName                lang="en">Related
People</displayName>
    <SearchType>nodeQurey</SearchType>
    <up>*</up>
    <dwon>1</down>
    <siblings>true</siblings>
```

【００７７】　　【表１２】

43 44

SIBLINGS='true'タグは、現オブジェクトと同じ親を有するすべてのオブジェクトが返されるべきであることを示す。<up>、<down>および<siblings>タグは、いかなる自動解析によっても生成されない。それらは、メタバース・デザイナによって注意深くデザインされ、クライアントがユーザに有益なビューを提供することを可能にする。

```
        <ElementsList>
            <element>cn</element>
            <element>uid</element>
            <element>telephoneNumber</element>
            <element>title</element>
            <element>buildingName</element>
            <element>roomNumber</element>
        </ElementsList>
    </view>
    <view>
        <displayName lang="en">Same Title (incontext)</displayName>
        <SearchType>nodeQuery</SearchType>
        <up>*</up>
        <SearchElement>title</SearchElement>
```

「サーチ要素(SearchElement)」タグは、クライアントに対して、どの関心を持たれる要素が、指定されたエンティティ上で照会されるべきであるかを示す。例えば、Jane Doeが選ばれ、「肩書」サーチ要素が指定された場合、クライアントは Jane Doe の肩書を判定し、その肩書を有するすべての人をサーチする。

```
        <ElementsList>
            <element>cn</element>
            <element>uid</element>
            <element>telephoneNumber</element>
            <element>title</element>
            <element>buildingName</element>
            <element>roomNumber</element>
        </ElementsList>
    </view>
    <view>
```

【００７８】

【表１３】

45 46

```
<displayName lang="en">Same Title (list)</displayName>
<SearchType>nodeSearch</SearchType>
<SearchElement>title</SearchElement>
<ElementsLis>
<element>cn</element>
<element>uid</element>
<element>telephoneNumber</element>
<element>title</element>
<element>buildingName</element>
<element>roomNumber</element>
</ElementsLis>
</view>
</dimension>
<dimension>
<name>officeLocation</name>
<upnodeReferenceElement>locationUpnode</upnodeReferenceElement>
<dimensionNamingElement>location</dimensionNamingElement>
<view>
<displayName lang="en">Location ofOffice</displayName>
<SearchType>nodeQuery</SearchType>
<up>*</up>
<ElementsLis>
<element>cn</element>
<element>uid</element>
<element>telephoneNumber</element>
<element>title</element>
<element>buildingName</element>
<element>roomNumber</element>
<element>description</element>
</ElementsLis>
</view>
</dimension>
</Dimensions>
<Inputs>
```

【００７９】

【表１４】

「入力（Inputs）」タグは、ポリアーキ・データ・マネージャに対して、もととなる材料がどこに位置するか、およびその材料に対してどの要素をアンカーとして用いるべきかを示す。

```
<Input name="base" path="input.xml" anchor="GlueID"/>
</Inputs>
</WellknownEntities>
```

【００８０】（データポリアーキを動的に生成する例示 50 的な手順）図４に、オブジェクトの属性の値に基づくオ

ブジェクト間関係の複数階層を生成する例示的な手順４００を示す。データポリアーキ１２２に、複数のオブジェクトが含まれる。この手順は、コンピュータ可読媒体に保管されたコンピュータ実行可能命令としてソフトウェアで実施し、媒体に機能的に結合されたプロセッサによって実行される時に、命令が、図４のブロックに示された動作を実行するものとすることができる。

【００８１】ブロック４１０で、図１のデータポリアーキサーバ１０２が、Ｘ－５００およびＬＤＡＰに基づく従来のディレクトリサービス、メタディレクトリサービス、データベースなどの、任意の数のデータソースからデータを受け取る。データは、ＸＭＬデータフォーマットなどの複数の異なるデータフォーマットのいずれかで受け取られる。サーバ１０２は、受け取ったデータを、図１のデータポリアーキ管理モジュール１２０に通信する。

【００８２】ブロック４１２で、データの受取（ブロック４１０）に応答して、データポリアーキ管理モジュール１２０が、受け取ったデータとポリアーキ１２２内に既にあるデータ（存在する場合に）の間のオブジェクト間関係（たとえば単一方向または両方向の関係）を反映するようにデータポリアーキ１２２を生成するか更新する。既に述べたように、これらのオブジェクト間関係は、ポリアーキ内の他のオブジェクトの属性に関する、受け取ったデータの属性に基づいて判定される。具体的に言うと、データポリアーキを生成、構成、または更新するために、管理モジュールは、データポリアーキ内のオブジェクトの属性の相対分布を分析して、０個、１個、または複数の次元のうちで、各オブジェクトがポリアーキ内の他のオブジェクトとのオブジェクト間関係に参加するものがどれであるかを判定する。

【００８３】これらの動作４１２は、データの受取（ブロック４１０）に自動的にまたは動的に応答し、ネットワーク管理者などの人間のオペレータの介入を必要としない。データポリアーキ１２２内のオブジェクト間関係が、ポリアーキ内のオブジェクトの属性の値に基づいて判定され、表現されるので、これらのオブジェクト間データ関係を、弾力性のあるものとすることができ、これは、それらのオブジェクト間データ関係が、経時的に変化することができることを意味する。属性の値が変化する際に、新しい値に基づくオブジェクト間関係が、受取時に管理モジュール１２０によってポリアーキ内で動的にまたは自動的に表現される。これらの動作４１２は、データポリアーキ内のデータオブジェクトの各々の１つの間のデータ関係の先見的知識から独立に実行される。さらに、データポリアーキ内のオブジェクト間関係が、ポリアーキ内のオブジェクトの属性の値に基づいて判定され、表現されるので、これらの関係は、オブジェクトの識別名から完全に独立に判定され、表現される。

【００８４】ブロック４１４で、図１のデータポリアーキ管理モジュール１２０が、関心を持たれる要素のスキーマ１２４の生成、構成、または更新を行って（たとえば図１および３を参照されたい）、データポリアーキ１２２を意味のある形で操作、提示、およびナビゲートできる方法を示す。具体的に言うと、図２および３と表７に関して上で説明したように、スキーマによって、データポリアーキ内の要素または属性が、各属性の対応する特徴的特性、位置決め特性、または分類特性と共に示される。スキーマによって、ポリアーキ内の次元も、その次元内のオブジェクトによって含まれる関心を持たれる要素または属性のそれぞれと共に示される。

【００８５】ブラウザによって、ポリアーキ的データセット１２２の検索、ナビゲート、または部分の表示に使用されるポリアーキ的照会言語（ＰＱＬ）コマンドの例示的セット（関心を持たれる要素のスキーマ１２４に基づく）を、図６から１２に関して下で詳細に説明する。関心を持たれる要素のスキーマ１２４を使用してＰＱＬ要求および応答を式で表す例示的な手順を、図１３に関して下で詳細に説明する。

【００８６】（例示的ポリアーキ的照会言語要求）図５に、データポリアーキ内の情報に対応する情報（ポリアーキ的照会言語（ＰＱＬ）応答）を返すようにデータポリアーキサーバ１０２に要求するためにクライアント１１０によって使用される例示的ＰＱＬ照会を示す。そのような照会の受取に応答して、データポリアーキ管理モジュール１２０が、ポリアーキ内のオブジェクトに対応する情報の組を識別し、取り出す。

【００８７】照会５００および対応するサーバ１０２の応答は、ＸＭＬなどのテキストマークアップ言語を使用して実施される。この構成では、照会およびサーバ応答が、シンプル・オブジェクト・アクセス・プロトコル（ＳＯＡＰ：Ｓｉｍｐｌｅ　Ｏｂｊｅｃｔ　Ａｃｃｅｓｓ　Ｐｒｏｔｏｃｏｌ）でパッケージ化され、ハイパーテキスト・トランスファ・プロトコル（ＨＴＴＰ：Ｈｙｐｅｒｔｅｘｔ　Ｔｒａｎｓｆｅｒ　Ｐｒｏｔｏｃｏｌ）を使用して図１のネットワーク１０４上にポストされる。ＳＯＡＰおよびＨＴＴＰは、ネットワーク通信プロトコルの当業者に周知の通信プロトコルである。

【００８８】メッセージ５００には、スキーマＩＤ５０２と、１つ以上の図２の属性２１４を指定するための１つ以上のオブジェクト変換パラメータ５１０（以下では、パラメータを、データフィールドとも称する）が含まれる。スキーマＩＤは、特定の関心を持たれる要素のスキーマ１２４を識別するのに使用される。このデータフィールドは、デフォルトスキーマまたは１つのスキーマだけがある場合には任意選択であることを諒解されたい。属性５１０は、データポリアーキ１２２の仮想オブジェクト２１０に対応する（属性には、特徴的属性、位置決め属性、または分類属性が含まれ、これらのそれぞれを、上で図２の論理ドメインプロパティに関して詳細

に説明した）。

【００８９】パラメータ５１０またはデータフィールドは、そのタイプに従って分類され、タイプは、特定の関心を持たれる要素タイプ５１０－１、応答を制限するための関心を持たれる要素修飾子５１０－２、論理修飾子５１０－３、次元インジケータ５１０－４、または次元情報修飾子５１０－５を含むタイプから選択される。メッセージ５００で表現されるデータフィールドの数およびタイプは、メッセージの設計または目的に基づく。

【００９０】図６に、例示的なＰＱＬ照会５００メッセージと、対応する例示的なＰＱＬ応答６２０とを表示するユーザインターフェース（ＵＩ）６００を示す。具体的には、ＰＱＬ照会に、データポリアーキ１２２の検索を実行するのに用いる特定の属性５１０を指定するための、データポリアーキスキーマ１２４に基づく修飾子パラメータ５１０－１が含まれる。ＵＩには、ＰＱＬメッセージ５００を入力するための第１区域６１０、ＳＯＡＰエンベロープ６１８にパッケージ化され、ＨＴＴＰを介してポストされるＰＱＬメッセージを示す第２区域６１２、およびデータポリアーキ管理モジュール１２０のＰＱＬ応答６２０を示す第３区域６１４が含まれる。ＰＱＬ応答は、ＳＯＡＰエンベロープで返されるものとして図示されているが、応答を、さまざまな他のデータパッケージフォーマットで返すことができる。

【００９１】この例では、特定の関心を持たれる要素パラメータ５１０－１によって、姓属性「Ｄｏｅ」が指定される。ＰＱＬ応答６２０では、少なくとも２つのオブジェクトと、対応する関心を持たれる要素が返される。各々のＧｌｕｅ　ＩＤによって、各々のオブジェクトが識別され、これは特徴的要素である。第１のオブジェクトは、「Ｊｏｈｎ　Ｄｏｅ」に関係する。第２のオブジェクトは、「ＪｉｍＤｏｅ」に関係する。各オブジェクトは、部屋番号（「ｒｏｏｍ　ｎｕｍｂｅｒ」）、ユーザＩＤ（「ｕｉｄ」）、姓（「ｓｎ」）、名（「ｇｉｖｅｎ　ｎａｍｅ」）、建物名（「ｂｕｉｌｄｉｎｇ　ｎａｍｅ」）、肩書（「ｔｉｔｌｅ」）、関係する次元の表示（「ｌｏｃａｔｉｏｎＵｐｎｏｄｅ」）、エンティティマネージャ（「ｍａｎａｇｅｒ」）、コストセンタＩＤ（「ｃｏｓｔ　ｃｅｎｔｅｒ　ｕｐ　ｎｏｄｅ」）などの複数の関心を持たれる要素と共に返された。

【００９２】特定の関心を持たれる要素によって、データポリアーキ１２２に保管された特定のオブジェクトに対応するＧＵＩＤなどの絶対的に一意の特徴的属性が指定された場合には、サーバ１０２は、その特定のオブジェクトに関するデータポリアーキ１２２に保管された情報のすべてを返す。

【００９３】図７に、検索の結果を修正するのに用いられる制限属性を指定する、関心を持たれる要素修飾子データフィールド５１０－２を有する例示的ＱＰＬ照会を表示するユーザインターフェース６００を示す。制限属性は、ポリアーキ的データスキーマ１２４によって表されるオブジェクトの組に対応する。関心を持たれる要素修飾子データフィールドによって、サーバに、検索動作に対する応答が、制限属性に関して識別されたデータポリアーキ１２２オブジェクトの提示に制限されることが示される。

【００９４】この例では、制限属性５１０－２が、普通名（「ｃｎ」）属性および一意識別子属性である。したがって、サーバによって返されるさまざまな人物オブジェクト６２０では、これらの制限属性だけが示される。

【００９５】図８に、データ関係のポリアーキに関する数学的動作を実行する論理修飾子パラメータ５１０－３を含む例示的なＰＱＬ照会５００のユーザインターフェースを示す。論理修飾子は、変数に基づくデータ関係の１つ以上の階層に対してフィルタリング動作（「ａｎｄ」）、和集合動作（「ｏｒ」）、または排他動作（「ｎｏｔ」）を実行するのに使用される。変数には、図１および３と表７のデータポリアーキスキーマ１２４によって表されるオブジェクト、スキーマによって表されるオブジェクトの階層、またはスキーマによって表されるオブジェクトのポリアーキなどが含まれる。

【００９６】たとえば、「ａｎｄ」論理修飾子５１０－３は、特定の関心を持たれる要素データフィールド５１０－１に基づく２つのデータストア検索の結果をフィルタリングするのに使用される。第１の特定の関心を持たれる要素データフィールドでは、値が「Ｓｍｉｔｈ」の姓（「ｓｎ」）属性が指定される。第２の特定の関心を持たれる要素データフィールドでは、値が「副社長（ｖｉｃｅ　ｐｒｅｓｉｄｅｎｔ）」の「肩書（ｔｉｔｌｅ）」属性が指定される。したがって、論理修飾子を使用して、各々の検索結果に基づいて結果を狭めるかフィルタリングすることができる。結果は、副社長Ｊｏｈｎ　Ｓｍｉｔｈに対応する、ＰＱＬ応答６２０の単一のオブジェクトである。この照会５００に一致するエンティティ情報の複数の組がディレクトリに保管されている場合には、エンティティのそれぞれが、結果に提示される。

【００９７】図９に、データストアに保管されたオブジェクトの検索動作に対応する応答をその中で提示する次元を指定するための次元情報インジケータデータフィールド５１０－４を含む、例示的なＰＱＬ照会５００を示すユーザインターフェース６００を示す。この例では、「下（ｕｎｄｅｒ）」パラメータ５１０－４（または「句」）が、フィルタ５１０－３（「＜ａｎｄ＞」）と組み合わされて、「１２３４５６７８９８」という対応する「一意識別子（ｕｎｉｑｕｅ　ｉｄｅｎｔｉｆｉｅｒ）」を有することが示されている、Ｊｏｈｎ　Ｓｍｉｔｈの下の「構造（ａｒｃｈｉｔｅｃｔ）」が検索される（図８のＪｏｈｎ　Ｓｍｉｔｈの一意識別子も参照されたい）。Ｊｏｈｎ　Ｓｍｉｔｈの下の構造に対応する

情報が、サーバ１０２からのＰＱＬ応答６２０に提示される（関心を持たれる要素データフィールド５１０－２が、検索の結果に提示される要素の数を制限するのに使用される方法に留意されたい）。

【００９８】図１０に、次元情報修飾子データフィールド５１０－５を有するＰＱＬ照会５００を示すユーザインターフェース６００を示す。次元情報修飾子によって、ポリアーキ的スキーマ内の複合オブジェクトと１つ以上の異なる表現されたオブジェクトの間のデータ関係を提示するための特定の方向および特定の深さが指定される。方向は、１つ以上の（「＊」などのワイルドカード指示を使用する場合はすべてのオブジェクト）異なるオブジェクトが、複合オブジェクトのサブオブジェクトであるかどうかを示す。次元情報修飾子では、ＳＩＢＬＩＮＧＳ（兄弟）＝’ｔｒｕｅ’を指定して、現在のオブジェクトと同一の親を有するすべてのオブジェクトを返さなければならないことを示すこともできる。

【００９９】この例では、次元情報修飾子５１０－５が、第１レベルの従属する者１０１０に対応する情報６２０をデータストアから取り出すのに使用される。これは、ジャンプゲートである。というのは、Ｊｏｈｎ Ｓｍｉｔｈに従属する者１０１０が、Ｊｏｈｎ Ｓｍｉｔｈのオブジェクト定義６２０の態様として提示されるからである。

【０１００】図１１は、特定の属性と、データ関係の２つのポリアーキの間の後続の交差とに関するＰＱＬ照会５００でのフィルタパラメータの使用を示す、ユーザインターフェース６００のブロック図である。この例では、２つの次元５１０－４（たとえば、「マネージメント（ｍａｎａｇｅｍｅｎｔ）」次元と「オフィスの位置（ｏｆｆｉｃｅ ｌｏｃａｔｉｏｎ）」次元）が、交差され、「構造（ａｒｃｈｉｔｅｃｔ）」の「肩書（ｔｉｔｌｅ）」属性に基づいてフィルタリングされる５１０－３。検索結果６２０に、その照会に一致する、データストア内の特定のオブジェクトが示される。

【０１０１】図１２に、データ関係の２つのポリアーキ５１０－４の間のフィルタ（「ａｎｄ」）５１０－３および和集合（「ｏｒ」）５１０－３を指定するＰＱＬ５００を示すユーザインターフェース６００が示されている。この例では、フィルタおよび和集合が、論理修飾子である。和集合属性は、「マネージメント（ｍａｎａｇｅｍｅｎｔ）」次元および「オフィスの位置（ｏｆｆｉｃｅ ｌｏｃａｔｉｏｎ）」次元に適用される。フィルタでは、「構造（ａｒｃｈｉｔｅｃｔ）」の「肩書（ｔｉｔｌｅ）」属性が指定され、これは、２つの階層の和集合に適用される。検索結果６２０に、この照会に一致する、データストア内の特定のオブジェクトが示される。

【０１０２】（ポリアーキ的データセットを管理する例示的手順）図１３に、データポリアーキ１２２内のデータを管理する例示的な手順１３００を示す。ブロック１３１０で、ポリアーキ的データ管理モジュール１２０が、関心を持たれる要素のスキーマ１２４をクライアント１１０に通信する。関心を持たれる要素のスキーマ１２４によって、クライアントに、クライアントが意味のある形でデータポリアーキ内のオブジェクトのアクセス、操作、および提示を行える方法が示される。

【０１０３】ブロック１３１２で、ポリアーキ的データ管理モジュール１２０が、通信されたデータポリアーキスキーマ（ブロック１３１０）に基づくＰＱＬメッセージ５００を受け取る。この要求では、関心を持たれる１つ以上の属性が識別されるだけではなく、関心を持たれるデータ関係も識別される。この要求は、図１のポリアーキ的データセット１２２内のデータオブジェクト群の１データオブジェクトに対応する。

【０１０４】受け取られたＰＱＬメッセージ５００は、（ａ）データオブジェクトのデフォルト検索オブジェクトを見つける動作、（ｂ）データオブジェクト群のうちで特定の名前に対応するオブジェクトを突き止める動作、（ｃ）データオブジェクト群の１特定のオブジェクトに対応するデータ関係のデフォルト階層を得る動作、（ｄ）データオブジェクト群の１特定のオブジェクトに対応するデータ関係の特定の階層を得る動作、（ｅ）データオブジェクト群の１特定のオブジェクトに対応するデータ関係の複数の階層の少なくとも１つのサブセットを識別する動作、（ｆ）データオブジェクト群の１特定のオブジェクトに対応するデータ関係の複数階層を得る動作、などの任意の組み合わせを含む１つ以上の動作に対応するものとすることができる。

【０１０５】ブロック１３１４で、データポリアーキ管理モジュール１２０が、物理アクセスストラテジ（たとえば単純スキャン、再帰スキャンなど）を判定して、要求に対応するデータをデータポリアーキ１２２から識別する。この判定は、要求（ブロック１３１２）に基づき、要求は、クライアント１１０に通信された（ブロック１３１０）スキーマ１２４に基づく。既に注記したように、スキーマは、クライアントに、データポリアーキの可能な内容に対応する情報を与えるだけではなく、関心を持たれるいずれかのオブジェクトに関係する可能性があるデータ関係の可能なポリアーキを記述した情報も含む（たとえば、表７に示された「＜Ｄｉｍｅｎｓｉｏｎ（次元）＞」インジケータを参照されたい）。

【０１０６】たとえば、クライアント要求（すなわちＰＱＬメッセージ５００）が、絶対的に一意の特徴的属性（たとえばＧＵＩＤおよびそのＧＵＩＤに対応する普通名）を除く、オブジェクトに関係する関心を持たれる要素のすべてをフィルタアウトするように設計される場合に、データポリアーキ１２２の単純スキャンが、関心を持たれる特徴的オブジェクトに関する情報に関する検索の効率的な技法であることを考慮されたい。

53

【０１０７】しかし、要求５００で、複合オブジェクト（すなわちジャンプゲート）に関して複数のサブオブジェクトを提示しなければならず、その後、結果を、１つ以上のサブオブジェクトに直交する複合オブジェクトに対応する情報の次元の和集合によって後に修正しなければならないことを示すこともできる。この場合には、データポリアーキ１２２の再帰スキャンが、関心を持たれるオブジェクトおよびオブジェクト間関係に関する情報の検索に効率的な技法である。

【０１０８】この形で、関心を持たれる属性およびデータ関係を識別するＰＱＬ要求メッセージ５００が、そのような属性およびデータ関係についてデータポリアーキ１２２を検索する最適化された物理アクセスストラテジも提供する。

【０１０９】ブロック１３１６で、データポリアーキ管理モジュール１２０が、判定された物理アクセスストラテジ（ブロック１３１４）に基づいて、ポリアーキからデータにアクセスする。アクセスされるデータは、複数の異なる形態をとることができる。たとえば、アクセスされるデータは、データオブジェクトとポリアーキ内の他のオブジェクトの間のオブジェクト間関係のいずれからも独立とすることができる。さらに、アクセスされるオブジェクトが、ポリアーキ内の１つ以上の異なるデータオブジェクトとのオブジェクト間関係の１つ以上の階層に参加することができる。この場合には、アクセスされるオブジェクトおよび１つ以上の異なるオブジェクトに、類似する属性が含まれる。上で述べたように、これらのオブジェクト間関係は、１つ以上の次元でお互いに関して直交するものとすることができる。

【０１１０】ブロック１３１８で、ポリアーキ的データ管理モジュール１２０が、クライアント１１０への発行のために、アクセスされたデータを変換する。具体的に言うと、アクセスされたデータは、データの要求に使用された（ブロック１３１２）特定のＰＱＬメッセージ５００の要件に基づいて変換される。たとえば、メッセージで、特定の次元に関してオブジェクトが示される場合に、アクセスされたデータの暗示的および明示的なオブジェクト間関係が、特定の次元に基づいて階層に組み立てられる。

【０１１１】たとえば、アクセスされたデータオブジェクトは、アクセスされたデータに、データオブジェクトの１つ以上のサブオブジェクトに関係するデータオブジェクトの複合オブジェクトが含まれる時に、ジャンプゲートを表す。この場合には、複合オブジェクトが、独立のサーフェスエンティティとして変換または表現される。１つ以上のサブオブジェクトのそれぞれが、サーフェスエンティティに独立の形で各々の別々のエンティティとして記述される。その後、１つ以上のサブオブジェクトが、サーフェスエンティティ内で変換または参照されて、複合オブジェクトと１つ以上のサブオブジェクトの間の関係が示される。参照は、すべてのオブジェクト命名または複合オブジェクトと１つ以上のサブオブジェクトの間の階層データ関係から独立である。

54

【０１１２】もう１つの例では、アクセスされたデータに、１つ以上のサブオブジェクトに関係する、ポリアーキ内のデータオブジェクトのうちの第１オブジェクトが含まれる。第１オブジェクトは、独立のサーフェスエンティティとして変換または表現される。１つ以上のサブオブジェクトのそれぞれが、サーフェスエンティティから独立の形で各々の別々のエンティティとして記述される。その後、各々のリンクが、第１オブジェクトを参照するために、１つ以上のサブオブジェクトのそれぞれに含められる。この形で、前の例のように、データが変換されて、対応するＰＱＬメッセージ５００で示される、関心を持たれる関係が表現される。

【０１１３】ブロック１３２０で、データポリアーキ管理モジュール１２０が、変換されたデータ（ブロック１３１８）をクライアントに発行または通信する。

【０１１４】（例示的なコンピューティング環境）図１４に、図１の例示的なデータポリアーキサーバ１０２を実施できる適するコンピューティング環境１４００の例を示す。この例示的なコンピューティング環境は、適するコンピューティング環境の１例にすぎず、例示的なデータポリアーキサーバ１０２、サーバ１０６、またはクライアント１１０の使用または機能性の範囲に関する制限を暗示することは意図されていない。コンピューティング環境１４００が、例示的なコンピューティング環境１４００に図示された構成要素のいずれかまたはその組合せに関する依存性または要件を有すると解釈してもならない。

【０１１５】コンピュータ１４０２は、多数の他の汎用のまたは特殊目的のコンピューティングシステム環境またはコンピューティングシステム構成と共に作動可能である。例示的なコンピュータ１４０２と共に使用するのに適する可能性がある周知のコンピューティングシステム、コンピューティング環境、またはコンピューティング構成の例には、パーソナルコンピュータ、サーバコンピュータ、シンクライアント、シッククライアント、ハンドヘルドデバイス、ラップトップデバイス、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラム可能消費者電子機器、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、上記のシステムまたはデバイスのいずれかを含む分散コンピューティング環境、および類似物が含まれるが、これに制限されない。

【０１１６】例示的なコンピュータ１４０２を、コンピュータによって実行される、プログラムモジュールなどのコンピュータ実行可能命令の一般的なコンテキストで説明することができる。一般に、プログラムモジュールには、特定のタスクを実行するか特定の抽象データ型を

実装する、ルーチン、プログラム、オブジェクト、コンポーネント、データ構造などが含まれる。例示的なコンピュータ１４０２は、通信ネットワークを介してリンクされたリモート処理デバイスによってタスクが実行される分散コンピューティング環境で実践することができる。分散コンピューティング環境では、プログラムモジュールを、メモリストレージデバイスを含む、ローカルおよびリモートの両方のコンピュータ記憶媒体に配置することができる。

【０１１７】図１４からわかるように、コンピューティング環境１４００には、コンピュータ１４０２の形の汎用コンピューティングデバイスが含まれる。コンピュータ１４０２の構成要素には、１つ以上のプロセッサまたは処理ユニット１４１２、システムメモリ１４１４、および、システムメモリ１４１４を含むさまざまなシステム構成要素をプロセッサ１４１２に結合するバス１４１６を含めることができるが、これに制限されない。

【０１１８】バス１４１６は、メモリバスまたはメモリコントローラ、周辺バス、加速されたグラフィック・ポート（ａｃｃｅｌｅｒａｔｅｄ　ｇｒａｐｈｉｃｓ　ｐｏｒｔ）、および、さまざまなバスアーキテクチャのいずれかを使用するプロセッサバスまたはローカルバスを含む、複数のタイプのバス構造のいずれかの１つ以上を表す。制限ではなく例として、そのようなアーキテクチャには、インダストリ・スタンダード・アーキテクチャ（ＩＳＡ：Ｉｎｄｕｓｔｒｙ　Ｓｔａｎｄａｒｄ　Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、マイクロチャネルアーキテクチャ（ＭＣＡ）バス、拡張ＩＳＡ（ＥＩＳＡ）バス、ビデオ・エレクトロニクス・スタンダード・アソシエーション（ＶＥＳＡ：Ｖｉｄｅｏ　Ｅｌｅｃｔｒｏｎｉｃｓ　Ｓｔａｎｄａｒｄｓ　Ａｓｓｏｃｉａｔｉｏｎ）ローカルバス、および、メザニンバスとも称するペリフェラル・コンポーネント・インターコネクト（ＰＣＩ：Ｐｅｒｉｐｈｅｒａｌ　Ｃｏｍｐｏｎｅｎｔ　Ｉｎｔｅｒｃｏｎｎｅｃｔｓ）バスが含まれる。

【０１１９】サーバ１４０２には、通常は、さまざまなコンピュータ可読媒体が含まれる。そのような媒体は、コンピュータ１４０２によってアクセス可能なすべての使用可能な媒体とすることができ、これには、揮発性媒体および不揮発性媒体と、取外し可能媒体および取外し不能媒体の両方が含まれる。

【０１２０】図１４では、システムメモリ１４１４に、ランダムアクセスメモリ（ＲＡＭ）１４２０などの揮発性メモリ、または読取専用メモリ（ＲＯＭ）１４１８などの不揮発性メモリ、もしくはその両方の形の、コンピュータ可読媒体が含まれる。起動中などにコンピュータ１４０２内の要素の間で情報を転送するのを助ける基本ルーチンを含む基本入出力システム（ＢＩＯＳ）１４２２が、ＲＯＭ１４１８に保管される。ＲＡＭ１４２０には、通常は、プロセッサ１４１２によって即座にアクセス可能または現在操作されているデータまたはプログラムモジュールが含まれる。

【０１２１】コンピュータ１４０２には、さらに、他の取外し可能／取外し不能、揮発性／不揮発性のコンピュータ記憶媒体を含めることができる。例のみとして、図１４に、取外し不能で不揮発性の磁気媒体（図示せず。通常は「ハードドライブ」と呼ばれる）を読み書きするためのハードディスクドライブ１４２４、取外し可能で不揮発性の磁気ディスク１４２８（たとえば「フロッピディスク」）を読み書きするための磁気ディスクドライブ１４２６、ＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭ、または他の光学媒体などの取外し可能で不揮発性の光ディスク１４３２を読み書きするための光ディスクドライブ１４３０が示されている。ハードディスクドライブ１４２４、磁気ディスクドライブ１４２６、および光ディスクドライブ１４３０は、それぞれ、１つ以上のインターフェース１４３４によってバス１４１６に接続される。

【０１２２】ドライブおよびそれに関連するコンピュータ可読媒体は、コンピュータ読取可能な命令、データ構造、プログラムモジュール、およびコンピュータ１４０２用の他のデータの不揮発性記憶域を提供する。本明細書に記載の例示的環境ではハードディスク、取外し可能磁気ディスク１４２８、および取外し可能光ディスク１４３２を使用するが、磁気カセット、フラッシュメモリカード、ディジタルビデオディスク、ランダムアクセスメモリ（ＲＡＭ）、読取専用メモリ（ＲＯＭ）および類似物などの、コンピュータによってアクセス可能であるデータを保管することができる他のタイプのコンピュータ可読媒体も、例示的オペレーティング環境内で使用可能であることを、当業者は諒解するであろう。

【０１２３】複数のプログラムモジュール１４４０を、ハードディスク、磁気ディスク１４２８、光ディスク１４３２、ＲＯＭ１４１８、またはＲＡＭ１４２０に保管することができ、このプログラムモジュールには、制約ではなく例として、オペレーティングシステム１４３８、１つ以上のアプリケーションプログラム１４４０、他のプログラムモジュール１４４２、およびプログラムデータ１４４４が含まれる。

【０１２４】そのようなオペレーティングシステム１４３８、１つ以上のアプリケーションプログラム１４４０（たとえばポリアーキデータ管理モジュール１２０）、他のプログラムモジュール１４４２、およびプログラムデータ１４４４（たとえばデータポリアーキ１２２および関心を持たれる要素のスキーマ１２４）のそれぞれ、またはそれらのある組合せに、図１の例示的なデータポリアーキサーバ１０２の実施形態を含めることができる。具体的に言うと、これらのそれぞれに、下記のためにデータポリアーキサーバ１０２の実施形態を含めることができる。

【０１２５】（ａ）オブジェクトの属性値に基づくデー

タボリアーキ１２２の動的な生成、管理、および更新、（ｂ）データボリアーキ内のオブジェクトをさまざまなオブジェクト間関係内で意味のある形で提示でき操作できる方法を示す、関心を持たれる要素のスキーマを生成するための、属性の相対分布に基づくデータボリアーキの分析、（ｃ）関心を持たれる要素のスキーマ１２４のクライアント１１０への通信、（ｄ）スキーマに基づく照会（たとえばＰＱＬメッセージ５００）の受取に応答する、ボリアーキ的データセット１２２から要求されたデータにアクセスするための物理アクセスストラテジの判定、（ｅ）照会要求に基づくデータのアクセスおよび変換、および（ｆ）応答としてのクライアントへの変換されたデータの発行。

【０１２６】ユーザは、キーボード１４４６およびポインティングデバイス１４４８（「マウス」など）などの任意選択の入力装置を介して、コンピュータ１４０２にコマンドおよび情報を入力することができる。他の入力装置（図示せず）には、マイクロホン、ジョイスティック、ゲームパッド、衛星アンテナ、シリアルポート、スキャナ、および類似物を含めることができる。これらおよび他の入力装置は、バス１４１６に結合されるユーザ入力インターフェース１４５０を介して処理ユニット１４１２に接続されるが、パラレルポート、ゲームポート、またはユニバーサルシリアルバス（ＵＳＢ）などの他のインターフェース構造およびバス構造によって接続することができる。

【０１２７】任意選択のモニタ１４５２または他のタイプの表示装置も、ビデオアダプタ１４５４などのインターフェースを介してバス１４１６に結合される。モニタのほかに、パーソナルコンピュータには、通常は、出力周辺インターフェース１４５５を介して接続することができる、スピーカおよびプリンタなどの他の周辺出力装置（図示せず）が含まれる。

【０１２８】コンピュータ１４０２は、リモートサーバ／コンピュータ１４６２（たとえばデータサーバ１０６）などの１つ以上のリモートコンピュータへの論理接続を使用して、ネットワーク環境で動作することができる。リモートコンピュータ１４６２には、コンピュータ１４０２に関して本明細書で説明する要素および特徴の多くまたはすべてを含めることができる。

【０１２９】図１４に示された論理接続は、ローカルエリアネットワーク（ＬＡＮ）１４５７および一般的な広域ネットワーク（ＷＡＮ）１４５９である。そのようなネットワーク環境は、オフィス、企業内コンピュータネットワーク、イントラネット、およびインターネットでありふれたものである。ＬＡＮネットワーキング環境で使用される時に、コンピュータ１４０２は、ネットワークインターフェースまたはネットワークアダプタ１４６６を介してＬＡＮ１４５７に接続される。ＷＡＮネットワーキング環境で使用される時に、コンピュータには、通常、ＷＡＮ１４５９を介する通信を確立するための、モデム１４５８または他の手段が含まれる。モデムは、内蔵または外付けとすることができるが、ユーザ入力インターフェース１４５０または他の適当な機構を介してシステムバス１４１６に接続することができる。

【０１３０】図１４には、インターネットを介するＷＡＮの特定の実施形態が図示されている。コンピュータ１４０２には、通常、インターネット１４６０を介する通信を確立するための、モデム１４５８または他の手段が含まれる。モデムは、内蔵または外付けとすることができるが、インターフェース１４５０を介してバス１４１６に接続される。

【０１３１】ネットワーク化された環境では、コンピュータ１４０２に関して図示されたプログラムモジュールまたはその一部を、リモートメモリストレージデバイスに保管することができる。制限ではなく例として、図１４に、リモートコンピュータ１４６２のメモリデバイスに常駐するリモートアプリケーションプログラム１４６９を示す。図示され説明されたネットワーク接続が、例示的であり、コンピュータ間の通信リンクを確立する他の手段を使用できることを諒解されたい。

【０１３２】（コンピュータ可読媒体）例示的なコンピュータ１０２の実施形態を、ある形のコンピュータ可読媒体に保管するか、それにまたがって伝送することができる。コンピュータ可読媒体は、コンピュータによってアクセス可能なすべての使用可能な媒体とすることができる。制限ではなく例として、コンピュータ可読媒体に、「コンピュータ記憶媒体」および「通信媒体」を含めることができる。

【０１３３】「コンピュータ記憶媒体」には、コンピュータ可読命令、データ構造、プログラムモジュール、または他のデータなどの情報の保管のためのあらゆる方法またはテクノロジで実施された、揮発性および不揮発性の、取外し可能および取外し不能の媒体が含まれる。コンピュータ記憶媒体には、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリまたは他のメモリテクノロジ、ＣＤ－ＲＯＭ、ディジタル多用途ディスク（ＤＶＤ）または他の光学ストレージ、磁気カセット、磁気テープ、磁気ディスクストレージまたは他の磁気記憶装置、もしくは、所望の情報を保管するのに使用でき、コンピュータによってアクセスできる他のすべての媒体が含まれるが、これに制限されない。

【０１３４】「通信媒体」は、通常は、搬送波または他のトランスポート機構などの変調されたデータ信号で、コンピュータ可読命令、データ構造、プログラムモジュール、または他のデータを実施する。通信媒体には、すべての情報配布媒体も含まれる。

【０１３５】用語「変調されたデータ信号」は、信号内に情報をエンコードする形でその特性の１つ以上をセットされるか変更された信号を意味する。制限ではなく例

59

として、通信媒体に、有線ネットワークまたは直接配線接続などの有線媒体と、音響、ＲＦ、赤外線、および他の無線媒体などの無線媒体が含まれる。上記のいずれかの組み合わせも、コンピュータ可読媒体の範囲に含まれる。

【０１３６】（結論）説明した構成および手順によって、単一の静的階層内でオブジェクト間関係を表現する識別名という従来の概念が置換される。具体的に言うと、説明した構成および手順によって、これらの従来の概念が、オブジェクトの属性に基づくデータ関係の複数次元でのオブジェクト間接続の動的に生成されるグラフに置換される。この形で、複雑な実世界のオブジェクトが、特定のオブジェクト自体に関して、特定のオブジェクトに関係する分解されたサブエンティティまたはサブオブジェクトの組に関して表現される。これらのオブジェクト間関係は、データポリアーキ１２２内の関心を持たれる要素にアクセスするために生成されたデータポリアーキスキーマ１２４を使用して管理され、ナビゲートされる。

【０１３７】データ関係のポリアーキを生成し管理するための、説明した内容は、構造的特徴または方法論的動作に固有の言語で説明されたが、請求項で定義される主題が、必ずしも説明された特定の特徴または動作に制限されないことを理解されたい。そうではなく、特定の特徴およびステップは、請求される本発明を実施する好ましい形態として開示されたものである。

【０１３８】

【発明の効果】本発明によれば、オブジェクト属性値に基づくオブジェクト間関係の複数階層の動的生成が提供される。

【図面の簡単な説明】

【図１】オブジェクトの属性の値に基づいてオブジェクト間関係の複数階層を動的に生成し管理する例示的システムを示す図である。

【図２】オブジェクトの属性の値に基づく動的に生成されるオブジェクト間関係の複数階層を表す例示的なポリアーキデータ構造を示す図である。

【図３】図２のデータポリアーキを意味のある形で作成し、アクセスし、操作する方法を示す、例示的なスキーマデータ構造を示す図である。

【図４】オブジェクトの属性の値に基づくオブジェクト間関係の複数階層を生成する例示的手順を示す図である。

【図５】データポリアーキから情報（ポリアーキ的照会言語（ＰＱＬ）応答）を返すようにサーバに要求するためにクライアントによって使用される例示的ＰＱＬ要求を示す図である。

【図６】例示的なＰＱＬ照会と、対応する例示的なＰＱＬ応答を表示するユーザインターフェース（ＵＩ）を示す図であり、具体的には、ＰＱＬ照会に、ポリアーキ的

60

データセットの検索を実行するのに用いる特定の属性を指定するための、データポリアーキスキーマに基づく修飾子パラメータが含まれる図である。

【図７】例示的なＰＱＬ照会と、対応する例示的なＰＱＬ応答を表示するＵＩを示すブロック図であり、具体的には、ＰＱＬ照会に、関心を持たれる要素のスキーマに基づく修飾子パラメータが含まれ、このパラメータによって、検索の結果を修正するのに用いられる制限属性が指定されるブロック図である。

【図８】例示的なＰＱＬ照会と、対応する例示的なＰＱＬ応答を表示するＵＩを示すブロック図であり、具体的には、ＰＱＬ照会に、データ関係のポリアーキに関する数学的動作を実行する論理修飾子パラメータが含まれるブロック図である。

【図９】例示的なＰＱＬ照会と、対応する例示的なＰＱＬ応答を示すＵＩのブロック図であり、具体的には、ＰＱＬ照会に、データストアに保管されたオブジェクトをその中で表示する次元を指定するための次元情報インジケータパラメータが含まれるブロック図である。

【図１０】例示的なＰＱＬ照会と、対応する例示的なＰＱＬ応答を示すＵＩを示す図であり、具体的には、ＰＱＬ照会に、次元情報修飾子パラメータが含まれ、これによって、サーバプロセスが、ポリアーキ的データセットの複合オブジェクトと１つ以上の他のオブジェクトの間のデータ関係を提示するための、特定の階層方向および特定の階層深さが指定される図である。

【図１１】例示的なＰＱＬ応答を形成するための、特定の属性およびデータ関係の２つの対応するポリアーキの間の後続の交差に関する例示的なＰＱＬ照会での位置決め要素の使用を示すＵＩを示す図である。

【図１２】例示的なＰＱＬ照会と、対応する例示的なＰＱＬ応答を示すＵＩを示す図であり、具体的には、ＰＱＬ照会に、データ関係の２つのポリアーキに関するフィルタパラメータおよび和集合パラメータの使用が示されている図である。

【図１３】データポリアーキ（すなわち、オブジェクトの属性の値に基づく動的に生成されるオブジェクト間関係の複数階層）内のデータを管理する（たとえば、オブジェクトにアクセスする、オブジェクトを提示する、提供する、操作するなど）例示的な手順の諸態様を示す図である。

【図１４】データポリアーキを管理する例示的なオペレーティング環境の諸態様を示す図である。

【符号の説明】

１０２　データポリアーキサーバ
１０４　ネットワーク
１０６　データサーバ
１０８－１～１０８－４　データベース
１１０　クライアントコンピュータ
１１２　プロセッサ

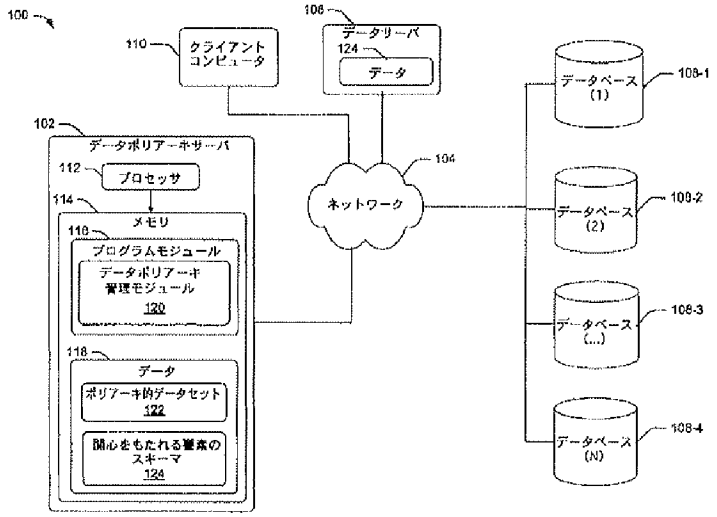
61

１１４　メモリ
１１６　プログラムモジュール
１１８　データ
１２０　データポリアーキ管理モジュール
１２２　ポリアーキ的データセット
１２４　関心を持たれる要素のスキーマ
２１０－１～２１０－Ｎ　仮想オブジェクト
２１２　ＧＵＩＤデータフィールド
２１４　属性データフィールド
２１６　述部データフィールド
２１８　ドメインプロパティデータフィールド
２２０　サブオブジェクトエンティティ参照
３１０－１　特徴的要素
３１０－２　位置決め要素
３１０－３　分類要素
５００　ポリアーキ的データセット照会
５０２　スキーマＩＤ
５１０　オブジェクト変換パラメータ
５１０－１　特定の関心を持たれる要素
５１０－２　応答を制限するための修飾子
５１０－３　論理修飾子
５１０－４　次元情報インジケータ
５１０－５　次元情報修飾子
６００　ユーザインターフェース
６１０　第１区域
６１２　第２区域
６１４　第３区域
６１８　ＳＯＡＰエンベロープ
６２０　ＰＱＬ応答
１４００　コンピューティング環境

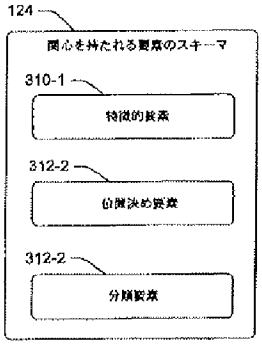
62

１４１２　処理ユニット
１４１４　システムメモリ
１４１６　バス
１４１８　ＲＯＭ
１４２０　ＲＡＭ
１４２２　ＢＩＯＳ
１４２４　ハードディスクドライブ
１４２６　磁気ディスクドライブ
１４２８　磁気ディスク
１４３０　光ディスクドライブ
１４３２　光ディスク
１４３４　データ媒体インターフェース
１４３８　オペレーティングシステム
１４４０　アプリケーションプログラム
１４４２　他のプログラムモジュール
１４４４　プログラムデータ
１４４６　キーボード
１４４８　ポインティングデバイス
１４５０　ユーザ入力インターフェース
１４５２　モニタ
１４５４　ビデオアダプタ
１４５５　出力周辺インターフェース
１４５７　ＬＡＮ
１４５８　モデム
１４５９　ＷＡＮ
１４６０　インターネット
１４６２　リモートコンピュータ
１４６６　ネットワーク
１４６９　リモートアプリケーション

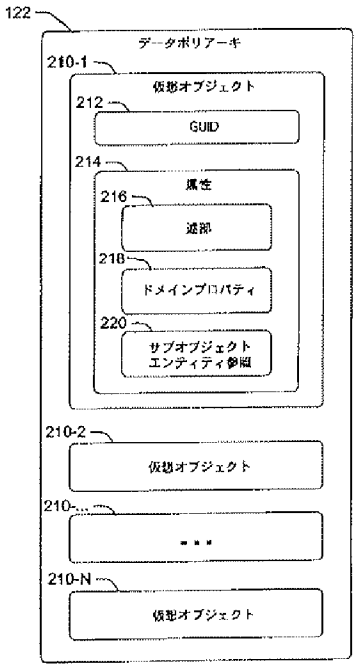
【図１】



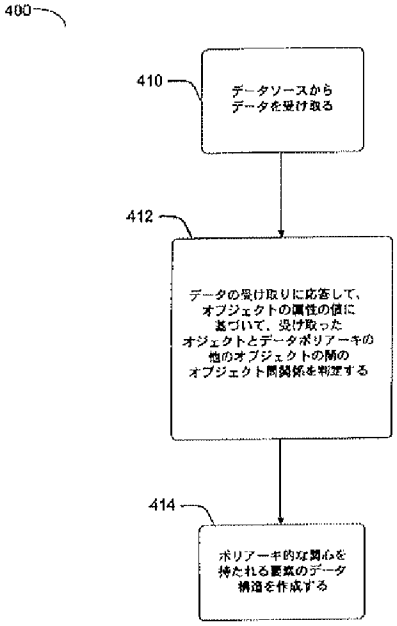
【図３】



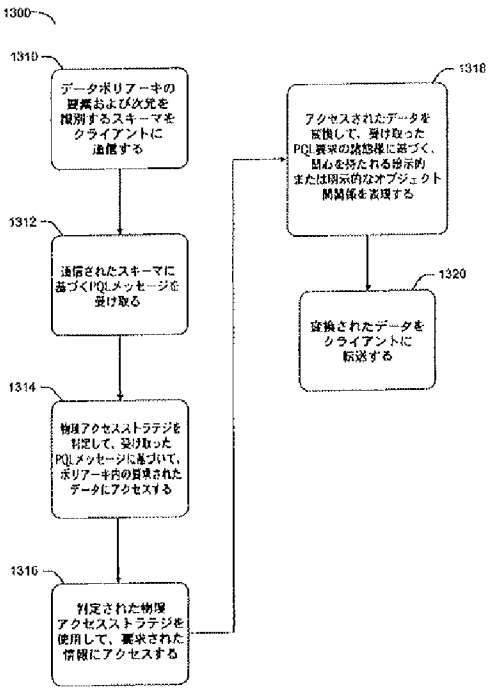
【図２】


122
データポリアーキ
210-1
仮想オブジェクト
212
GUID
214
属性
216
述部
218
ドメインプロパティ
220
サブオブジェクト
エンティティ参照
210-2
仮想オブジェクト
210-...
...
210-N
仮想オブジェクト

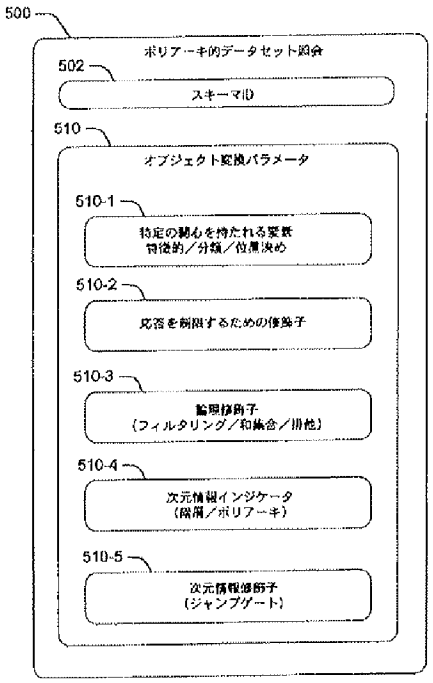
【図４】


400
410
データソースから
データを受け取る
412
データの受け取りに応答して、
オブジェクトの属性の値に
基づいて、受け取った
オブジェクトとデータポリアーキの
他のオブジェクトの間の
オブジェクト間関係を判定する
414
ポリアーキ的な関心を
持たれる要素のデータ
構造を作成する

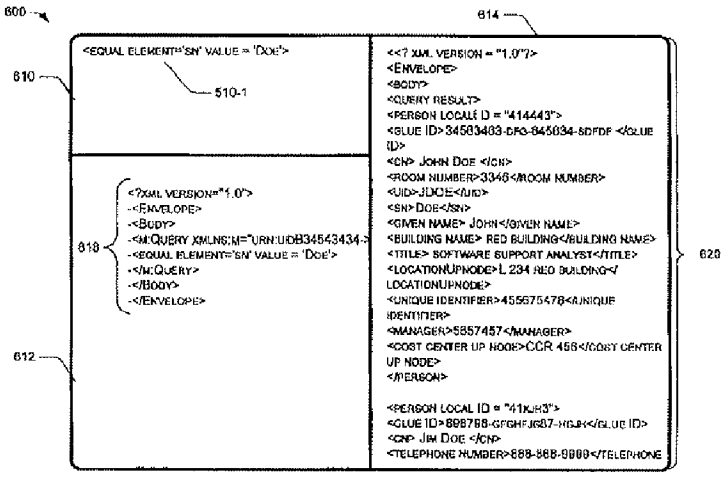
【図１３】


1300
1310
データポリアーキの
要素および次元を
識別するスキーマを
クライアントに
通信する
1312
通信されたスキーマに
基づくPQLメッセージを
受け取る
1314
物理アクセスストラテジを
判定して、受け取った
PQLメッセージに基づいて、
ポリアーキ内の要求された
データにアクセスする
1316
判定された物理
アクセスストラテジを
使用して、要求された
情報にアクセスする
1318
アクセスされたデータを
変換して、受け取った
PQL要求の述部に基づく、
関心を持たれる暗示的
または明示的なオブジェクト
間関係を実現する
1320
変換されたデータを
クライアントに
転送する

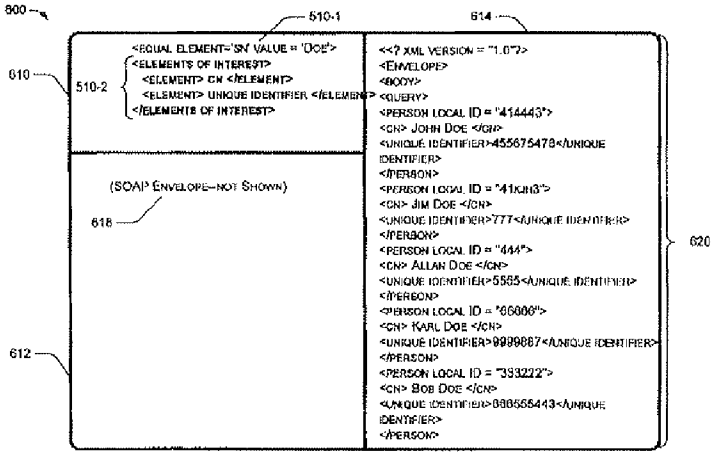
【図５】


500
ポリアーキ的データセット照会
502
スキーマID
510
オブジェクト変換パラメータ
510-1
特定の関心を持たれる要素
特徴的／分類／位置決め
510-2
応答を制限するための修飾子
510-3
制限修飾子
（フィルタリング／和集合／排他）
510-4
次元情報インジケータ
（階層／ポリアーキ）
510-5
次元情報修飾子
（ジャンプゲート）

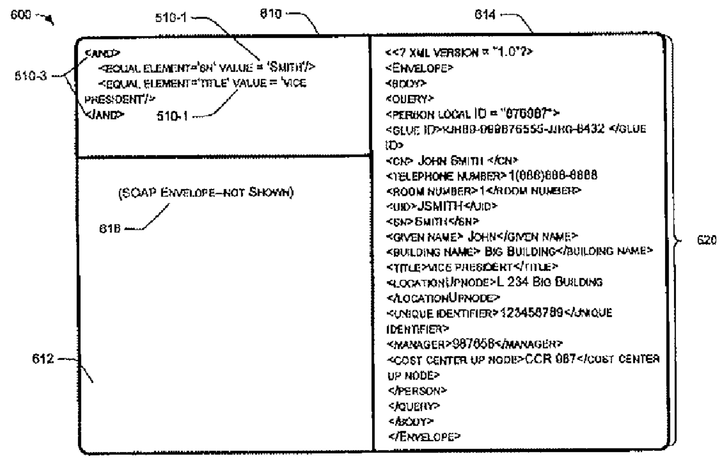
【図６】


600
614
<EQUAL ELEMENT='SN' VALUE = 'DOE'>
610
510-1
618
<?XML VERSION="1.0">
-<ENVELOPE>
-<BODY>
-<MQUERY XMLNS:M="URN:UID834543434">
-<EQUAL ELEMENT='SN' VALUE = 'DOE'>
-</M:QUERY>
-</BODY>
-</ENVELOPE>
612
<<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY RESULT>
<PERSON LOCALE ID = "414443">
<GLUE ID>34583403-DF3-845034-SDFDF </GLUE ID>
<CN> JOHN DOE </CN>
<ROOM NUMBER>3346</ROOM NUMBER>
<UID>JDOE</UID>
<SN>DOE</SN>
<GIVEN NAME> JOHN</GIVEN NAME>
<BUILDING NAME> RED BUILDING</BUILDING NAME>
<TITLE> SOFTWARE SUPPORT ANALYST</TITLE>
<LOCATIONUPNODE> L 234 RED BUILDING</LOCATIONUPNODE>
<UNIQUE IDENTIFIER>455675478</UNIQUE IDENTIFIER>
<MANAGER>5557457</MANAGER>
<COST CENTER UP NODE>CCR 456</COST CENTER UP NODE>
</PERSON>
<PERSON LOCAL ID = "41KJH3">
<GLUE ID>898788-GFGHFJG87-HGJH</GLUE ID>
<CN> JIM DOE </CN>
<TELEPHONE NUMBER>888-888-9999</TELEPHONE
620

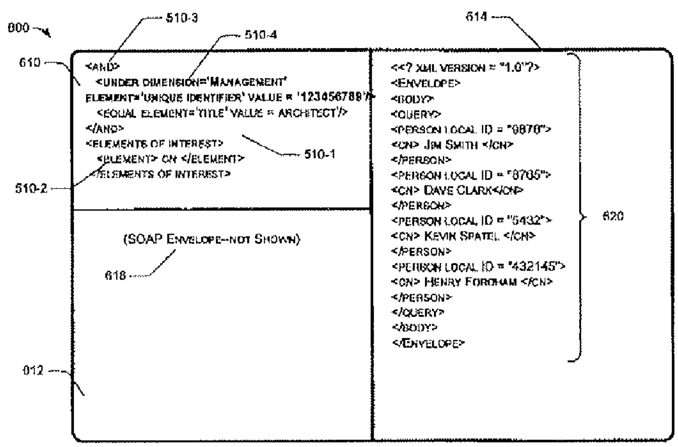
【図７】


600
510-1
614
610
510-2
<EQUAL ELEMENT='SN' VALUE = 'DOE'>
<ELEMENTS OF INTEREST>
<ELEMENT> CN </ELEMENT>
<ELEMENT> UNIQUE IDENTIFIER </ELEMENT>
</ELEMENTS OF INTEREST>
(SOAP ENVELOPE–NOT SHOWN)
618
612
<<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY>
<PERSON LOCAL ID = "414443">
<CN> JOHN DOE </CN>
<UNIQUE IDENTIFIER>455675478</UNIQUE IDENTIFIER>
</PERSON>
<PERSON LOCAL ID = "41KJH3">
<CN> JIM DOE </CN>
<UNIQUE IDENTIFIER>777</UNIQUE IDENTIFIER>
</PERSON>
<PERSON LOCAL ID = "444">
<CN> ALLAN DOE </CN>
<UNIQUE IDENTIFIER>5565</UNIQUE IDENTIFIER>
</PERSON>
<PERSON LOCAL ID = "66666">
<CN> KARL DOE </CN>
<UNIQUE IDENTIFIER>9999667</UNIQUE IDENTIFIER>
</PERSON>
<PERSON LOCAL ID = "393222">
<CN> BOB DOE </CN>
<UNIQUE IDENTIFIER>666555443</UNIQUE IDENTIFIER>
</PERSON>
620

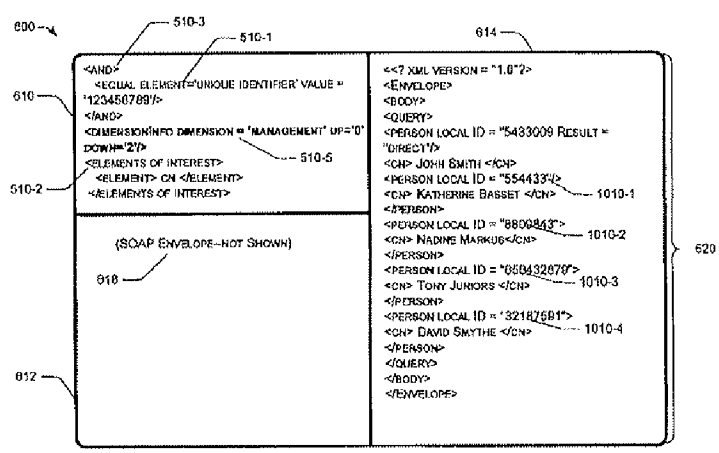
【図８】


600
510-1
610
614
<AND>
<EQUAL ELEMENT='SN' VALUE = 'SMITH'/>
<EQUAL ELEMENT='TITLE' VALUE = 'VICE PRESIDENT'/>
</AND>
510-3
510-1
(SOAP ENVELOPE--NOT SHOWN)
618
612
<<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY>
<PERSON LOCAL ID = "876887">
<GLUE ID>KJHB8-089876555-JJKG-8432 </GLUE ID>
<CN> JOHN SMITH </CN>
<TELEPHONE NUMBER>1(888)888-8888
<ROOM NUMBER>1</ROOM NUMBER>
<UID>JSMITH</UID>
<SN>SMITH</SN>
<GIVEN NAME> JOHN</GIVEN NAME>
<BUILDING NAME> BIG BUILDING</BUILDING NAME>
<TITLE>VICE PRESIDENT</TITLE>
<LOCATIONUPNODE>L 234 BIG BUILDING </LOCATIONUPNODE>
<UNIQUE IDENTIFIER>123456789</UNIQUE IDENTIFIER>
<MANAGER>987858</MANAGER>
<COST CENTER UP NODE>CCR 987</COST CENTER UP NODE>
</PERSON>
</QUERY>
</BODY>
</ENVELOPE>
620

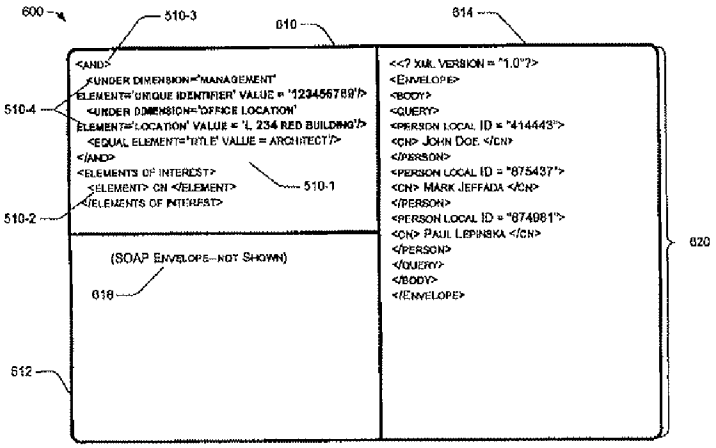
【図９】


600
510-3
510-4
614
610
<AND>
<UNDER DIMENSION='MANAGEMENT' ELEMENT='UNIQUE IDENTIFIER' VALUE = '123456789'/>
<EQUAL ELEMENT='TITLE' VALUE = ARCHITECT'/>
</AND>
<ELEMENTS OF INTEREST>
<ELEMENT> CN </ELEMENT>
</ELEMENTS OF INTEREST>
510-1
510-2
(SOAP ENVELOPE--NOT SHOWN)
618
612
<<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY>
<PERSON LOCAL ID = "9876">
<CN> JIM SMITH </CN>
</PERSON>
<PERSON LOCAL ID = "8765">
<CN> DAVE CLARK</CN>
</PERSON>
<PERSON LOCAL ID = "5432">
<CN> KEVIN SPATEL </CN>
</PERSON>
<PERSON LOCAL ID = "432145">
<CN> HENRY FORDHAM </CN>
</PERSON>
</QUERY>
</BODY>
</ENVELOPE>
620

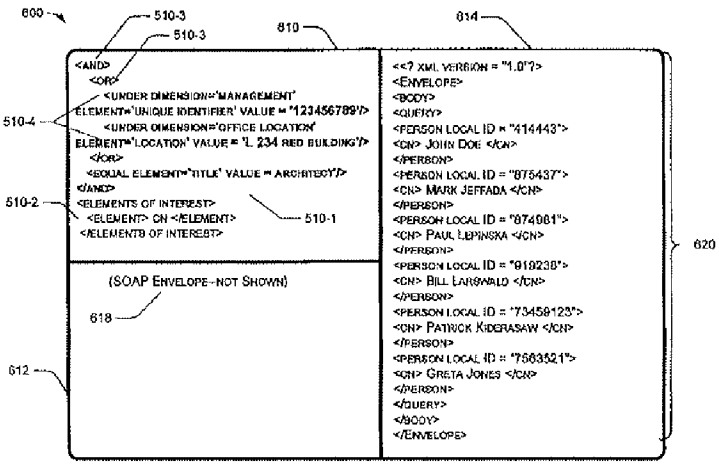
【図１０】


600
510-3
510-1
614
610
<AND>
<EQUAL ELEMENT='UNIQUE IDENTIFIER' VALUE = '123456789'/>
</AND>
<DIMENSIONINFO DIMENSION = 'MANAGEMENT' UP='0' DOWN='2'/>
510-5
<ELEMENTS OF INTEREST>
<ELEMENT> CN </ELEMENT>
</ELEMENTS OF INTEREST>
510-2
(SOAP ENVELOPE--NOT SHOWN)
618
612
<<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY>
<PERSON LOCAL ID = "5433009 RESULT = "DIRECT"/>
<CN> JOHN SMITH </CN>
<PERSON LOCAL ID = "554433"/>
<CN> KATHERINE BASSET </CN>
1010-1
</PERSON>
<PERSON LOCAL ID = "8809843">
1010-2
<CN> NADINE MARKUS</CN>
</PERSON>
<PERSON LOCAL ID = "850432870">
<CN> TONY JUNIORS </CN>
1010-3
</PERSON>
<PERSON LOCAL ID = "32187691">
<CN> DAVID SMYTHE </CN>
1010-4
</PERSON>
</QUERY>
</BODY>
</ENVELOPE>
620

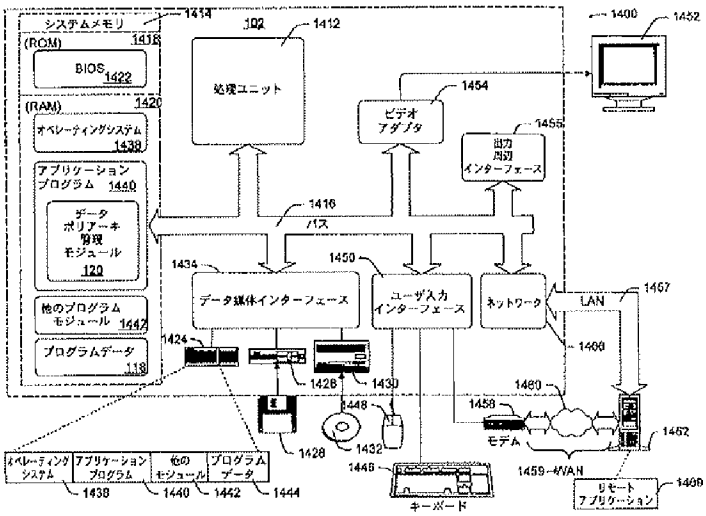
【図１１】


600
510-3
610
614
<AND>
<UNDER DIMENSION='MANAGEMENT'
ELEMENT='UNIQUE IDENTIFIER' VALUE = '123456789'/>
<UNDER DIMENSION='OFFICE LOCATION'
ELEMENT='LOCATION' VALUE = 'L 234 RED BUILDING'/>
<EQUAL ELEMENT='TITLE' VALUE = ARCHITECT/>
</AND>
<ELEMENTS OF INTEREST>
<ELEMENT> CN </ELEMENT>
</ELEMENTS OF INTEREST>
510-4
510-2
510-1
(SOAP ENVELOPE--NOT SHOWN)
618
612
<<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY>
<PERSON LOCAL ID = "414443">
<CN> JOHN DOE </CN>
</PERSON>
<PERSON LOCAL ID = "875437">
<CN> MARK JEFFADA </CN>
</PERSON>
<PERSON LOCAL ID = "874981">
<CN> PAUL LEPINSKA </CN>
</PERSON>
</QUERY>
</BODY>
</ENVELOPE>
620

【図１２】


600
510-3
510-3
610
614
<AND>
<OR>
<UNDER DIMENSION='MANAGEMENT'
ELEMENT='UNIQUE IDENTIFIER' VALUE = '123456789'/>
<UNDER DIMENSION='OFFICE LOCATION'
ELEMENT='LOCATION' VALUE = 'L 234 RED BUILDING'/>
</OR>
<EQUAL ELEMENT='TITLE' VALUE = ARCHITECT/>
</AND>
<ELEMENTS OF INTEREST>
<ELEMENT> CN </ELEMENT>
</ELEMENTS OF INTEREST>
510-4
510-2
510-1
(SOAP ENVELOPE--NOT SHOWN)
618
612
<<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY>
<PERSON LOCAL ID = "414443">
<CN> JOHN DOE </CN>
</PERSON>
<PERSON LOCAL ID = "875437">
<CN> MARK JEFFADA </CN>
</PERSON>
<PERSON LOCAL ID = "874981">
<CN> PAUL LEPINSKA </CN>
</PERSON>
<PERSON LOCAL ID = "919238">
<CN> BILL LARKWALD </CN>
</PERSON>
<PERSON LOCAL ID = "73459123">
<CN> PATRICK KIDERASAW </CN>
</PERSON>
<PERSON LOCAL ID = "7563521">
<CN> GRETA JONES </CN>
</PERSON>
</QUERY>
</BODY>
</ENVELOPE>
620

【図１４】



フロントページの続き

(72)発明者　キム　キャメロン
アメリカ合衆国　98004　ワシントン州　ベルビュー　サウスイースト　ショアライン　ドライブ　9328
(72)発明者　ジョージ　ジー．ロバートソン
アメリカ合衆国　98105　ワシントン州　シアトル　49　ノースイースト　38030
(72)発明者　マーク　アール．ブラウン
アメリカ合衆国　98112　ワシントン州　シアトル　マルデン　アベニュー　イースト　516
Ｆターム(参考)　5B075 KK02 ND35 NK46 NR02 NR20 QT03
5B082 EA01

【外国語明細書】

１．Title of the Invention

DYNAMICALLY GENERATING MULTIPLE HIERARCHIES OF INTER-OBJECT RELATIONSHIPS BASED ON OBJECT ATTRIBUTE VALUES

２．Claims

1. In a distributed computing environment, a method comprising:

receiving data from a data store, the data corresponding to a plurality of objects; and

responsive to receiving the data, dynamically generating multiple hierarchies of inter-object relationships based on values of attributes of the objects, the multiple hierarchies of inter-object relationships being a data polyarchy.

2. A method as recited in claim 1, wherein the data store comprises a directory or a database.

3. A method as recited in claim 1, wherein the data polyarchy comprises intersecting hierarchies of inter-object relationships.

4. A method as recited in claim 1, wherein the data polyarchy comprises an elastic inter-object relationship.

5. A method as recited in claim 1, wherein dynamically generating multiple hierarchies of inter-object relationships further comprises:

identifying a dimensional relationship of one or more dimensional relationships between a first and second object of the objects; and

inserting the first object into the second object such that the first object is represented in the second object with respect to the dimensional relationship.

6. A method as recited in claim 1, wherein first and second objects of the objects are respectively represented in the data polyarchy as separate entities, and wherein dynamically generating multiple hierarchies of inter-object relationships further comprises:

identifying a dimensional relationship of one or more dimensional relationships between the first object and the second object; and

inserting a link to the first object in the second object with respect to the dimensional relationship.

7. A method as recited in claim 6, wherein the link is a jump gate.

8. A method as recited in claim 1, wherein the multiple hierarchies of inter-object relationships are represented independent of object naming and independent of a predetermined hierarchical data structure.

9. A method as recited in claim 1, wherein the inter-object relationships represent mono-directional object relationships and bi-directional object relationships.

10. A method as recited in claim 1, wherein is the data polyarchy comprises a membership hierarchy that provides for de-referenced dimensional navigation of a many-to-many object relationship.

11. A method as recited in claim 1, wherein generating the data polyarchy further comprises:

relating a first and a second object of the objects to a third object of the objects to facilitate de-referenced dimensional navigation of a many-to-many object relationship between the first, second, and third objects.

12. A method as recited in claim 1, further comprising naming an inter-object relationship in the data polyarchy with a natural language.

13. A method as recited in claim 1, wherein generating the data polyarchy further comprises establishing, for individual ones of the objects, a plurality of predicates to indicate how to access the individual ones of the objects.

14. A method as recited in claim 1, wherein generating the data polyarchy further comprises establishing for individual ones of the objects a plurality of domain properties to index the individual ones of the objects.

15. A method as recited in claim 14, wherein the domain properties comprise a data type, a data precision indication, a scale indication, and a nullability indication.

16. A method as recited in claim 1, wherein generating the data polyarchy further comprises determining the relative distribution of attributes of the objects to establish a strategy to present or search for objects that comprise the attributes.

17. A method as recited in claim 1, wherein generating the data polyarchy further comprises:

determining the relative distribution of attributes of the objects to establish a strategy to present or search for objects that comprise the attributes, and wherein the strategy comprises one or more of the following operations:

a first operation to find a default search object of the objects;

a second operation to locate a particular object of the objects;

a third operation to obtain a default hierarchy of data relationships that correspond to a particular object of the objects;

a fourth operation to obtain a particular hierarchy of data relationships that correspond to a particular object of the objects;

a fifth operation to identify at least one subset of a plurality of hierarchies of data relationships that correspond to a particular object of the objects; and

a sixth operation to obtain multiple hierarchies of data relationships that correspond to a particular object of the objects.

18. A method as recited in claim 17, wherein the strategy comprises a recursive access strategy or a linear scan access strategy.

19. A method as recited in claim 17, wherein the domain properties comprise a logical domain property comprising a distinguishing domain, a locating domain, or a classifying domain.

20. A method as recited in claim 1, wherein each object further comprises one or more respective attributes, and wherein generating the data polyarchy further comprises:

identifying a plurality of distinguishing attributes, each distinguishing attribute representing a respective object of the objects that is a root of a hierarchy, each distinguishing attribute being from a substantially unique distribution of similar attributes across the objects;

identifying one or more locating attributes for narrowing a search for an object of the objects; each locating attribute being from a relatively large distribution of similar attributes across the objects; and

identifying one or more classifying attributes for filtering out objects from a search for an object, each classifying attribute being from a relatively small distribution of similar attributes across the objects.

21. A computer for representing directory-based object inter-object relationships, the computer comprising:

a processor; and

a memory coupled to the processor, the memory comprising computer-executable instructions and data, the processor for fetching and executing the computer-executable instructions, the computer-executable instructions comprising instructions for:

receiving data from a data store, the data corresponding to a plurality of objects; and

responsive to receiving the data, dynamically generating multiple hierarchies of inter-object relationships based on values of attributes of the objects, the multiple hierarchies of inter-object relationships being a data polyarchy.

22. A computer as recited in claim 21, wherein the data store comprises a directory or a database.

23. A computer as recited in claim 21, wherein the data polyarchy comprises intersecting hierarchies of inter-object relationships.

24. A computer as recited in claim 21, wherein the data polyarchy comprises an elastic inter-object relationship.

25. A computer as recited in claim 21,wherein the computer-executable instructions for dynamically generating multiple hierarchies of inter-object relationships further comprise instructions for:

identifying a dimensional relationship of one or more dimensional relationships between a first and second object of the objects; and

inserting the first object into the second object such that the first object is represented in the second object with respect to the dimensional relationship.

26. A computer as recited in claim 21, wherein first and second objects of the objects are respectively represented in the data polyarchy as separate entities, and wherein the computer-executable instructions for dynamically generating multiple hierarchies of inter-object relationships further comprise instructions for:

identifying a dimensional relationship of one or more dimensional relationships between the first object and the second object; and

inserting a link to the first object in the second object with respect to the dimensional relationship.

27. A computer as recited in claim 26, wherein the link is a jump gate.

28. A computer as recited in claim 21, wherein the multiple hierarchies of inter-object relationships are represented independent of object naming and independent of a predetermined hierarchical data structure.

29. A computer as recited in claim 21, wherein the inter-object relationships represent mono-directional object relationships and bi-directional object relationships.

30. A computer as recited in claim 21, wherein is the data polyarchy comprises a membership hierarchy that provides for de-referenced dimensional navigation of a many-to-many object relationship.

31. A computer as recited in claim 21, wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for:

relating a first and a second object of the objects to a third object of the objects to facilitate de-referenced dimensional navigation of a many-to-many object relationship between the first, second, and third objects.

32. A computer as recited in claim 21, wherein the computer-executable instructions for generating the data polyarchy further comprises instructions for establishing, for individual ones of the objects, a plurality of predicates to indicate how to access the individual ones of the objects.

33. A computer as recited in claim 21, wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for establishing for individual ones of the objects a plurality of domain properties identify to index the individual ones of the objects.

34. A computer as recited in claim 33, wherein the domain properties comprise a data type, a data precision indication, a scale indication, and a nullability indication.

35. A computer as recited in claim 21, wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for determining the relative distribution of attributes of the objects to establish a strategy to present or search for objects that comprise the attributes.

36. A computer as recited in claim 21, wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for:

determining the relative distribution of attributes of the objects to establish a strategy to present or search for objects that comprise the attributes, and wherein the strategy comprises one or more of the following operations:

a first operation to find a default search object of the objects;

a second operation to locate a particular object of the objects;

a third operation to obtain a default hierarchy of data relationships that correspond to a particular object of the objects;

a fourth operation to obtain a particular hierarchy of data relationships that correspond to a particular object of the objects;

a fifth operation to identify at least one subset of a plurality of hierarchies of data relationships that correspond to a particular object of the objects; and

a sixth operation to obtain multiple hierarchies of data relationships that correspond to a particular object of the objects.

37. A computer as recited in claim 36, wherein the strategy comprises a recursive access strategy or a linear scan access strategy.

38. A computer as recited in claim 36, wherein the domain properties comprise a logical domain property comprising a distinguishing domain, a locating domain, or a classifying domain.

39. A computer as recited in claim 21, wherein each object further comprises one or more respective attributes, and wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for:

identifying a plurality of distinguishing attributes, each distinguishing attribute representing a respective object of the objects that is a root of a hierarchy, each distinguishing attribute being from a substantially unique distribution of similar attributes across the objects;

identifying one or more locating attributes for narrowing a search for an object of the objects; each locating attribute being from a relatively large distribution of similar attributes across the objects; and

identifying one or more classifying attributes for filtering out objects from a search for an object, each classifying attribute being from a relatively small distribution of similar attributes across the objects.

40. A data structure comprising:

a plurality of virtual object data fields, each virtual object data field corresponding to a respective object of a plurality of objects in a data store, the

virtual object data fields indicating multiple hierarchies of inter-object relationships based on attributes of the objects.

41. A data structure as recited in claim 40, wherein the data store is a directory or a database.

42. A data structure as recited in claim 40, wherein each virtual object data field further comprises:

a first globally unique identifier (GUID) data field to uniquely identify a corresponding object in the data store.

43. A data structure as recited in claim 40, wherein a virtual object data field corresponds to a first object of the objects, and wherein the virtual object data field further comprises an entity reference data field to uniquely identify a second object of the objects as a sub-element of the first object, the entity reference data field uniquely identifying the second object in the data store.

44. A data structure as recited in claim 43, wherein the entity reference is a GUID.

45. A data structure as recited in claim 40, wherein each virtual data object data field further comprises one or more predicate data fields, each predicate data field indicating a respective operation to present a particular object with respect to one or more hierarchies of inter-object relationships.

46. A data structure as recited in claim 40, wherein each virtual data object data field further comprises:

a domain property data field to index a corresponding object of the objects with respect to one or more hierarchies of inter-object relationships.

47. A data structure as recited in claim 46, wherein the domain property data field further comprises:

a physical domain comprising a data type, a data precision indication, a scale indication, or a nullability indication; and

a logical domain comprising a unique domain, a locating domain, or a classifying domain.

48. A computer-readable medium having stored thereon a data structure as recited in claim 40.

49. A computer-readable medium comprising computer-executable instructions for:

receiving data from a data store, the data corresponding to a plurality of objects; and

responsive to receiving the data, dynamically generating multiple hierarchies of inter-object relationships based on values of attributes of the objects, the multiple hierarchies of inter-object relationships being a data polyarchy.

50. A computer-readable medium as recited in claim 49, wherein the data store comprises a directory or a database.

51. A computer-readable medium as recited in claim 49, wherein the data polyarchy comprises intersecting hierarchies of inter-object relationships.

52. A computer-readable medium as recited in claim 49, wherein the data polyarchy comprises an elastic inter-object relationship.

53. A computer-readable medium as recited in claim 49, wherein the data polyarchy comprises a complex object that is related to one or more sub-objects in the data polyarchy, and wherein the computer-executable instructions for determining inter-object relationships further comprise instructions for:

representing the complex object as an independent surface entity; and

referencing the one or more sub-objects in the independent surface entity as separate entities, the one or more sub-objects being referenced independent of object naming and independent of a hierarchical data relationship between the surface entity and the one or more sub-objects.

54. A computer-readable medium as recited in claim 49, wherein the data polyarchy comprises a first object that is related to one or more sub-objects in the data polyarchy, and wherein the computer-executable instructions for determining the inter-object relationships further comprise instructions for:

representing the first object as a surface entity;

representing each of the one or more sub-objects as respective separate entities that are independent of the surface entity; and

referencing the surface object in each of the one or more sub-objects independent of any object naming or hierarchical relationship.

55. A computer-readable medium as recited in claim 49, wherein the multiple hierarchies of inter-object relationships are represented independent of object naming and independent of a predetermined hierarchical data structure.

56. A computer-readable medium as recited in claim 49, wherein the inter-object relationships represent mono-directional object relationships and bi-directional object relationships.

57. A computer-readable medium as recited in claim 49, wherein is the data polyarchy comprises a membership hierarchy that provides for de-referenced dimensional navigation of a many-to-many object relationship.

58. A computer-readable medium as recited in claim 49, wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for:

relating a first and a second object of the objects to a third object of the objects to facilitate de-referenced dimensional navigation of a many-to-many object relationship between the first, second, and third objects.

59. A computer-readable medium as recited in claim 49, wherein the computer-executable instructions for generating the data polyarchy further comprises instructions for establishing, for individual ones of the objects, a plurality of predicates to indicate how to access the individual ones of the objects.

60. A computer-readable medium as recited in claim 49, wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for determining the relative distribution of attributes of the objects to establish a strategy to present or search for objects that comprise the attributes.

61. A computer-readable medium as recited in claim 49, wherein each object further comprises one or more respective attributes, and wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for:

identifying a plurality of distinguishing attributes, each distinguishing attribute representing a respective object of the objects that is a root of a hierarchy, each distinguishing attribute being from a substantially unique distribution of similar attributes across the objects;

identifying one or more locating attributes for narrowing a search for an object of the objects; each locating attribute being from a relatively large distribution of similar attributes across the objects; and

identifying one or more classifying attributes for filtering out objects from a search for an object, each classifying attribute being from a relatively small distribution of similar attributes across the objects.

62. A computer-readable medium as recited in claim 49, wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for establishing for individual ones of the objects a plurality of domain properties identify to index the individual ones of the objects.

63. A computer-readable medium as recited in claim 62, wherein the domain properties comprise a data type, a data precision indication, a scale indication, and a nullability indication.

64. A computer-readable medium as recited in claim 49, wherein the computer-executable instructions for generating the data polyarchy further comprise instructions for:

determining the relative distribution of attributes of the objects to establish a strategy to present or search for objects that comprise the attributes, and wherein the strategy comprises one or more of the following operations:

a first operation to find a default search object of the objects;

a second operation to locate a particular object of the objects;

a third operation to obtain a default hierarchy of data relationships that correspond to a particular object of the objects;

a fourth operation to obtain a particular hierarchy of data relationships that correspond to a particular object of the objects;

a fifth operation to identify at least one subset of a plurality of hierarchies of data relationships that correspond to a particular object of the objects; and

a sixth operation to obtain multiple hierarchies of data relationships that correspond to a particular object of the objects.

65. A computer-readable medium as recited in claim 64, wherein the strategy comprises a recursive access strategy or a linear scan access strategy.

66. A computer-readable medium as recited in claim 64, wherein the domain properties comprise a logical domain property comprising a distinguishing domain, a locating domain, or a classifying domain.

67. A computer for representing directory-based object inter-object relationships, the computer comprising processing means for:

receiving data from a data store, the data corresponding to a plurality of objects; and

responsive to receiving the data, dynamically generating multiple hierarchies of inter-object relationships based on values of attributes of the objects, the multiple hierarchies of inter-object relationships being a data polyarchy.

68. A computer as recited in claim 67, wherein the data polyarchy comprises intersecting hierarchies of inter-object relationships.

69. A computer as recited in claim 67, wherein the data polyarchy comprises an elastic inter-object relationship.

70. A computer as recited in claim 67, wherein the means for dynamically generating multiple hierarchies of inter-object relationships further comprise means for:

identifying a dimensional relationship of one or more dimensional relationships between a first and second object of the objects; and

inserting the first object into the second object such that the first object is represented in the second object with respect to the dimensional relationship.

71. A computer as recited in claim 67, wherein first and second objects of the objects are respectively represented in the data polyarchy as separate entities, and wherein the means for dynamically generating multiple hierarchies of inter-object relationships further comprise means for:

identifying a dimensional relationship of one or more dimensional relationships between the first object and the second object; and

inserting a link to the first object in the second object with respect to the dimensional relationship.

72. A computer as recited in claim 68, wherein the link is a jump gate.

73. A computer as recited in claim 67, wherein the multiple hierarchies of inter-object relationships are represented independent of object naming and independent of a predetermined hierarchical data structure.

74. A computer as recited in claim 67, wherein is the data polyarchy comprises a membership hierarchy that provides for de-referenced dimensional navigation of a many-to-many object relationship.

75. A computer as recited in claim 67, wherein the means for generating the data polyarchy further comprise means for:

relating a first and a second object of the objects to a third object of the objects to facilitate de-referenced dimensional navigation of a many-to-many object relationship between the first, second, and third objects.

76. A computer as recited in claim 67, wherein the means for generating the data polyarchy further comprises means for establishing, for individual ones of the objects, a plurality of predicates to indicate how to access the individual ones of the objects.

77. A computer as recited in claim 67, wherein the means for generating the data polyarchy further comprise means for establishing for individual ones of the objects a plurality of domain properties identify to index the individual ones of the objects.

78. A computer as recited in claim 77, wherein the domain properties comprise a data type, a data precision indication, a scale indication, and a nullability indication.

79. A computer as recited in claim 67, wherein the means for generating the data polyarchy further comprise means for determining the relative distribution of attributes of the objects to establish a strategy to present or search for objects that comprise the attributes.

80. A computer as recited in claim 67, wherein the means for generating the data polyarchy further comprise means for:

determining the relative distribution of values assumed by attributes of the objects to establish a strategy to present or search for objects that comprise the attributes, and wherein the strategy comprises one or more of the following operations:

a first operation to find a default search object of the objects;

a second operation to locate a particular object of the objects;

a third operation to obtain a default hierarchy of data relationships that correspond to a particular object of the objects;

a fourth operation to obtain a particular hierarchy of data relationships that correspond to a particular object of the objects;

a fifth operation to identify at least one subset of a plurality of hierarchies of data relationships that correspond to a particular object of the objects; and

a sixth operation to obtain multiple hierarchies of data relationships that correspond to a particular object of the objects.

81. A computer as recited in claim 80, wherein the strategy comprises a recursive access strategy or a linear scan access strategy.

82. A computer as recited in claim 80, wherein the domain properties comprise a logical domain property comprising a distinguishing domain, a locating domain, or a classifying domain.

83. A computer as recited in claim 67, wherein each object further comprises one or more respective attributes, and wherein the means for generating the data polyarchy further comprise means for:

identifying a plurality of distinguishing attributes, each distinguishing attribute representing a respective object of the objects that is a root of a hierarchy, each distinguishing attribute being from a substantially unique distribution of similar attributes across the objects;

identifying one or more locating attributes for narrowing a search for an object of the objects; each locating attribute being from a relatively large distribution of similar attributes across the objects; and

identifying one or more classifying attributes for filtering out objects from a search for an object, each classifying attribute being from a relatively small distribution of similar attributes across the objects.

RELATED APPLICATIONS

This application claims benefit of U.S. Provisional Application serial number 60/250,344 filed on November 30, 2000, which is hereby incorporated by reference.

TECHNICAL FIELD

The described subject matter relates to inter-object relationships. More particularly, the subject matter pertains to dynamically generating multiple hierarchies of inter-object relationships based on values of attributes of the objects.

BACKGROUND

Any object can be linked, correlated, associated, differentiated, or in some manner categorized with respect to a different object to form implicit or explicit inter-object relationships. For instance, in an organization, a person typically has implicit and explicit relationships with other people in the organization, organizational resources (e.g., printers, facilities, etc.), geographical locations, business units, club memberships, and so on. Each implicit and/or explicit relationship between respective objects (i.e., the person, the other people, a resource, etc.) represents a respective hierarchical data relationship.

For example, one hierarchical data relationship is represented by each person within the company that has access to a specific resource (e.g., a building on the company campus, a room, a printer, etc); the resource being the root node of the hierarchy and the individuals with access to the resource being the leaves. Another hierarchical data relationship is represented by individuals that make up the management structure of the company. Other inter-object data relationships

may represent a hierarchy of individuals within a particular business unit, all employees of the company that have specialized training, and so on.

Unfortunately, even though a data store can be configured to some extent by a network administrator to represent inter-object relationships within hierarchies of other data, complex inter-object relationships (e.g., such as those representing a single object within more that one hierarchy) are not simply and adequately represented using conventional data store (e.g., directory, database, etc.) systems and technologies. (Traditional directories include those based on the well-known X.500 standard and the Lightweight Directory Access Protocol (LDAP).

To illustrate this limitation of traditional data store systems and technologies, consider that a directory typically represents inter-object relationships using rigid data naming and inflexible directory schemas. Objects or nodes in the directory are organized within a single hierarchy with a root node at the top of the hierarchy. The root node has a name. Each other node in the directory is named based on its direct naming relationship to the root node and also with respect to each intervening node in the respective node's hierarchy. As a result, if a parent object is renamed in a single operation, any objects that are subordinate or children of the parent object are also renamed in that same single operation. This is because an object's full "distinguished name" includes the name of each parent object(s) all the way down the line to the root node's name.

It is the full distinguished name of an object that also represents its static location or data relationship with respect to each other object in the data store. Thus, an object's distinguished name inflexibly inter-tangles object naming within a single hierarchy with inter-object relationships in that hierarchy. Because of this,

any navigation of the data store must be performed from top-to-bottom to determine and subsequently present any inter-object relationships—that is from the root object, to a parent object to any subordinate child object(s).

Because traditional data stores (e.g., directories, databases, and so on) rely on a carefully specified and inflexible object naming scheme to identify inter-object relationships, an administrator configuring the data store requires a-priori knowledge of the inter-object relationships when configuring the data store. Additionally, any configuration of the data store must consider not only the proper representation of inter-object relationships in the data store, but must also consider the heuristics that a search engine requires to navigate the data store.

To make matters worse, elastic data relationships are not easily described, represented, or navigated using conventional data store systems and techniques. An elastic data relationship is one wherein the relationship is derived from data that defines an object at any point in time. This means that over time elastic data relationships can be dynamic. For instance, consider the following non-obvious and potentially elastic data relationships: a Web site and the Web pages that make up the Web site, a customer and the individuated services that the customer purchases from a merchant, a personal computer (PC) and peripheral devices that are coupled to the PC, a city and the districts within the city, a business and the business' contacts, an employee and the employee's dependents, and the like.

These non-obvious and potentially elastic data relationships are not easily represented because whenever a one-to-one correspondence between a surface object and corresponding sub-objects needs to be represented in the data store, an irreversible design choice must be made. (Conventional practice is to strictly control directory schema updates due to the serious nature of directory schema

modification). A network administrator can opt for "total incorporation" of the sub-objects into the particular object by representing the sub-objects as attributes of the surface object in the directory schema. Or the network administrator can opt for "total distinction" of each object, by creating separate objects in the schema for sub-object components, and positioning the separate objects subordinate to the surface object.

To illustrate this irreversible design choice, consider that a particular network router includes multiple router modules plugged into the router's backplane. Information about the router and the router modules are typically stored in a directory in one or two different fashions—each of which may be equally unsatisfactory depending on how entities and their respective relationships to other entities are represented. One design choice is to characterize a router and its corresponding router modules as a single hierarchical data structure representing the network router as a parent object, and the corresponding router modules as child objects that are subordinate to the parent object. A different design choice is to characterize the router and the router's associated router modules as a single parent object with complex attributes. The parent object represents the router (backplane), and the complex attributes represent the respective router modules that are hosted by the router.

In consideration of the first design choice, depending how the router and the modules are configured, collapsing information about the router modules, or boards onto the backplane may prove unwieldy. This is because the functionality of the router's backplane may be small as compared to the functionality of the network router modules hosted by the router. Whereas considering the second design choice, completely separating the boards from the backplane may be

equally unsatisfactory because the router is still a single physical router box that generally includes a number of router modules.

Both of the described solutions to representing data relationships with an inflexible directory schema are time consuming to implement and counter-intuitive. The semantics of shape and naming in the directory must be agreed on in advance to solve the simplest design problem. Thus, whenever a one-to-one correspondence between an entity and corresponding sub-entities needs to be represented in a traditional directory, an irreversible and inflexible design choice must be made within the directory schema.

Whichever design choice is selected, the data store and tools used to navigate, search and present objects within the data store with respect to inter-object relationships have been substantially limited. This is because the data store itself can not represent all of the possible implicit and explicit inter-object relationships of an object. This is considered by many computer programmers to be one of the most intractable problems of directory schema in traditional directories. This is also deemed to be the reason that computer program applications are not typically portable across directory platforms or even directory instances.

To further worsen matters, recent developments in information technology provide network administrators with opportunities to tie disparate data stores (e.g., databases, directories, and so on) of data together into a single logical directory or "metadirectory". Such disparate databases and directories include, for example, information corresponding to enterprise users, tangible and intangible resources, financial information, corporate e-mail systems, network operating systems, and so on.

Metadirectories present network administrators with complex and often elastic object data relationships that cannot be simply or adequately described, represented, navigated, or presented using traditional systems and procedures to configure and manage data stores. Considerable efforts are required on the part of the administrator (or a staff of administrators) to configure a data store. Manually determining and implementing such inter-object relationships (whether they be dynamic or not) is fraught with the potential for human error and oversight. Furthermore, database administrators with an appropriate level of such knowledge to perform such a directory configuration are expensive.

The following described subject matter addresses these and other problems of representing inter-object relationships.

SUMMARY

The described arrangements and procedures dynamically generate a data polyarchy from information received from a data store (e.g., a directory or database). The data polyarchy represents multiple hierarchies of inter-object relationships based on values of attributes of the objects. These multiple hierarchies are generated and represented in a manner that is independent of object naming and predetermined static hierarchical data structures.

DETAILED DESCRIPTION

Overview

The following subject matter replaces traditional notions of complex real-world object presentation within a single static hierarchy, wherein directory object naming and inter-object relationships are inter-tangled and unwieldy for representing complex data relationships. More specifically, traditional notions of distinguished names for representing inter-object relationships within a single directory of static inter-object relationships are replaced with graphs of elastic (non-static) inter-object connections in multiple dimensions of data relationships (e.g., mono and/or bi-directional relationships) based on attributes of the objects. In other words, the data relationships establish that one or more data objects participate in one or more respective dimensions, or polyarchies of inter-object relationships. One or more of these hierarchies can intersect creating intersecting hierarchies of inter-object relationships.

Dynamically generated multiple hierarchies of data relationships based on object attributes are represented in a data polyarchy. Specifically, the data polyarchy is generated using each object's respective data attributes or data values and multifarious interrelationships of those values with attributes that correspond to other objects in the polyarchical data set. The inter-object relationships in the data polyarchy can be elastic because inter-object relationships are derived from data defined by an object at any point in time. Patterns of relationships between objects emerge by presenting an object in one or more "dimensions" or polyarchies of data relationships. Such relationships are presented using inter-object connections between virtual entities representing the objects. A virtual

entity corresponds to an object of interest and includes and organizes information about an object of interest—including information about how to get more information about the object of interest. Such objects can be presented to people or computer programs that embody that interest.

In contrast to traditional systems and procedures to represent inter-object relationships in a data store, the following described arrangements and procedures are dynamic, in that they are automated and do not require any manual intervention from a network administrator to configure inter-object relationships. By dynamically generating a data polyarchy complex inter-object relationships based on object data are automatically determined without presenting any inflexible design choice to a schema designer.

This means that the network administrator or computer program (e.g., a search engine) is not required to have any a-priori knowledge of complex inter-object relationships to generate, navigate, or search a data store. This also means that each object in a data store can be viewed from as many different dimensional inter-object hierarchies as apply to the respective object. Furthermore, as an object's elastic data relationships change, the data polyarchy automatically detects and reflects those changes.

The following description sets forth arrangements and procedures based on a directory schema for representing polyarchies of inter-object relationships that incorporates elements recited in the appended claims. The subject matter is described with specificity to meet statutory requirements. However, the description itself is not intended to limit the scope of this patent. Rather, the inventors have contemplated that the claimed subject matter might also be embodied in other ways, to include different elements or combinations of elements

similar to the ones described in this document, in conjunction with other present or future technologies.

Exemplary System

Fig. 1 shows an exemplary system 100 to dynamically generate and manage multiple hierarchies of inter-object relationships based on the values of attributes of the objects. The system represents a distributed computing environment including a data polyarchy server 102 operatively coupled across a network 104 to one or more other optional data servers 106, one or more databases 108, and one or more client computers 110. The operative coupling of the data polyarchy server to the network can be made in any number of different ways such as through one or more server appliances (e.g., a server appliance on the outside of a Web farm-server farm), a corporate portal (intranet), a local area network (LAN), co-located a data store (e.g., a database 108), and so on.

The data polyarchy server 102 includes a processor 112 operatively coupled to a memory 114 that includes computer-executable instructions 116 and data 118. The processor is configured to fetch and execute the computer-executable instructions and fetch the data during such execution. Such computer-executable instructions include an operating system (not shown), and a data polyarchy management module 120 to dynamically generate and manage multiple hierarchies of inter-object relationships based on the values of attributes of the objects. These dynamically generated multiple hierarchies of inter-object relationships are stored in the polyarchical data set 122, which is also referred to as the data polyarchy. To generate the data polyarchy 118, the data polyarchy management module 120 uses data (e.g., Extensible Markup Language (XML)

data) from any number of different data sources such as from one or more other optional servers 106 and/or databases 108. For instance, a server 106 provides data (e.g., directories of enterprise users, resources, financial information, corporate e-mail systems, network operating systems, etc.) to the data polyarchy server from any number of various data stores—databases, directories, metadirectories, and so on. A database 108 is a structured or unstructured data store, including an object-oriented database such as an XML database, a Hypertext Markup Language (HTML) database, an SQL server database, and so on.

Responsive to generating and managing the data polyarchy 122, the management module 120 respectively generates and updates the elements of interest schema 124. The elements-of-interest schema indicates how an optional client computer 110 can manipulate and display the objects in the data polyarchy with respect to their respective polyarchies of inter-object relationships.

For instance, the elements-of-interest schema 124 identifies each object in the data polyarchy 122 as an address referencing a virtual entity (e.g., see the virtual object 210 of Fig. 2) that represents the respective object. These virtual entities are stored as vectors or arrays of addresses in the schema. Each different type of attribute that an object in the data polyarchy could have is also identified in the schema as well as what kinds of indexes are to be used on the various attribute types. (A data index provides for object access). For each attribute type it is convenient to store with its definition, its corresponding index. In this manner, for example, if somebody requests for an attribute, the index is readily available and all of the values assumed by the attribute can be determined very quickly. (An elements-of-interest schema is described in greater detail below in reference to Fig. 3).

The data polyarchy server 102 can generate any number of schemas 124. Each generated schema can provide access to various subsets of the objects in the data polyarchy 122 independent of the objects represented by other schemas 124. For example, a first schema 124 can be distributed to network administrators to provide access to resources and attributes such as printers and access lists that are otherwise protected or hidden from other employees. In the same manner, a second schema can be distributed to the president of human resources. While the second schema may provide the president with access to certain privileged employee records, the second schema could be completely silent with respect to the resources that are available to the network administrators group via the first schema. In this manner, schemas 124 can be designed to provide access control to organizational resources.

The data polyarchy server 102 communicates the elements-of-interest schema 124 to one or more optional clients 110. The client computer supports a graphical user interface (not shown) for displaying inter-object relationships in the data polyarchy 122 as described by the elements of interest schema. Exemplary arrangements and procedures to display objects within polyarchies of data relationships are described in related U.S. Patent application serial no. 09/728,935, titled "Hierarchy Polyarchy Visualization", filed on 11/29/00, which is assigned to the assignee hereof, and which is incorporated by reference.

The data polyarchy 122 and the elements of interest schema 124 can be replicated one or more times in a memory cache 114 by the data polyarchy server 102. An exemplary memory cache is described in greater detail below in reference to Fig. 14. Since the polyarchical server can operate either data set from a corresponding memory cache, there can be as many copies of the respective data

sets as necessary. Thus no matter how demanding a client 110, the data polyarchy server can satisfy the demand.

When data polyarchy 122 and the elements of interest schema 124 are replicated in a memory cache 114 by the data polyarchy server102, the server can maintain an authoritative store (not shown) in the memory 114 to represent the most recent, or current representation of the inter-object relationships. Such an authoritative store is beneficial because caches by their very nature are always out of date to some degree—meaning that data in a cache is only as "fresh", or timely as the most recent cache update. In light of this, a client requesting information from the data polyarchy 122 can indicate the level of data reliability or timeliness required by the client. If a high timeliness is required, the server 102 can access the data polyarchy from the authoritative store, rather than from more out of data caches. The speed of access to an authoritative cache depends on its respective implementation (e.g., implemented in internally to the server in random access memory or externally to the server in a data storage device).

Exemplary Data Polyarchy

Fig. 2 shows an exemplary polyarchical data set 122 to represent multiple dimensions of inter-object relationships based on attributes within the data. The data is anything that can be differentiated (e.g., anything that is an object of interest represented in a directory, database, etc., can be an object). The data set 122 is formatted to allow designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. For example, the data format can be an XML data format.

The data polyarchy 122 includes multiple virtual object data fields 210. Each virtual object data field includes and organizes information about a respective object, including, for example, information about how to get more information about the respective object. Specifically, the virtual object includes a globally unique identifier (GUID) data field 212 and if appropriate for the particular object, one or more attribute data fields 214.

A GUID 212 uniquely identifies the virtual object (which in turn represents a respective object) with respect to this or any other object in this or any other data polyarchy 122. As already noted, these objects can be represented in one or more physically distributed data stores that are in turn logically centralized by one or more directory services as well as by one or more data polyarchies. The attribute data field 214 defines any data attributes or data values of the virtual object 210. Each attribute corresponds to the attributes that an actual instance of the virtual object may include. Such attributes include, for example, one or more predicate data fields 216, multiple domain property data fields 218, and zero or more sub-object entity references 220.

Each predicate data field 216 indicates a respective operation to access or present a particular object with respect to one or more hierarchies of other objects (each object being represented by virtual objects 210 in the polyarchical data set 122). Such operations indicate one or more diverse types of searches (e.g., a linear search and a recursive search), data transformations (e.g., from one hierarchical relationship to another different hierarchical relationship), and so on. (See, block 1318 of Fig. 13).

If an object is a simple object, meaning that it does not reference to a sub-object entity 220, a predicate 216 operation (e.g., a search, modification, data

transformation—from one structure such as from a virtual object 210 to an object within a hierarchy of other objects) will correspond to the respective object of interest. However, if the object is a complex object, meaning that it has a data relationship to one or more sub-objects, then the predicate operation will correspond to a combination of the object and/or the one or more sub-objects.

The domain property data field 218 includes a physical domain property and a logical domain property. The physical domain property indicates one or more sets of values used to index a data object. The physical domain property is selected from a group of physical domain properties including a data type, a data precision indication, a scale indication, and a nullability indication). The logical domain property aspect of the domain property 218 facilitates searching and navigation of the data polyarchy 122 by allowing object data values to be assigned to particular domains. Specifically, the logical domain indicates a strategy to access and/or present the corresponding object with respect to the other objects in the data polyarchy. For instance, the logical domain property includes a unique domain property, a locating domain property, and a classifying domain property. The particular one logical domain property that the polyarchical data relationship management module 116 assigns to an attribute of an object is based on the attribute's relative distribution of its value in the data polyarchy with respect to other values of the same attribute of other objects in the data polyarchy.

We now describe: (a) the relative distribution of the values assumed by an attribute within the data polyarchy 122; and (b) how data distribution determines which objects represent respective dimensions (hierarchies), up-nodes, and down-nodes.

Relative Attribute Value Distribution

The set of values that an attribute has is part of that attribute's logical domain. Any information that is collected about the actual distribution of the values (in terms of the number of potential objects that contain each potential value) in a data polyarchy 111 is also a property of the attribute's logical domain. To determine the relative distribution of attribute values, one or more thresholds (e.g., a low threshold and a high threshold) are defined to determine the attribute's relative distribution in a data polyarchy 122 with respect to other attributes of other objects in the polyarchy. The thresholds are based on the assumption that the data may have a certain percentage of error within it (e.g., one (1) percent error). (Other statistical analysis techniques can be used in combination with or in place of the thresholds to determine object attribute distributions).

For instance, as objects are loaded into the data polyarchy 122 (e.g., from one or more directory and/or database servers 106), the data polyarchy management module 120 examines each object's respective attributes values based on the thresholds to determine: (a) which attributes are substantially unique with respect to their distributions in objects in the data set; (b) which attributes are distributed across a substantially large set of objects; and (c) which attributes are distributed across a substantially small set of objects in the data set. These determinations are made based on assuming that the data has that certain percentage of error.

With this assumption of some data error in mind, consider that a substantially unique attribute is not necessarily the only attribute of its kind in the data polyarchy 122. Rather, an attribute may be absolutely unique, or the attribute

may belong to a relatively sparse distribution of similar attributes in the data set. Attributes that are determined to be substantially unique with respect to their distributions across objects in the data set have a unique logical domain property illustrating that they are distinguishing as compared to other attributes.

Attributes that are distinguishing may identify respective unique dimensions in the polyarchical data set 122, which are represented as up-nodes of an interconnected graph that in turn represents a hierarchical dimension. Inside this model, the default polyarchy is flat. Attributes that are not distinguishing are distributed either across a substantially large set of objects in the data set, or alternatively distributed across a substantially small set of objects. Non-distinguishing attributes are not good candidates for attributes that define dimensions. Instead, such distributions indicate that non-distinguishing attributes belong to one or more of the identified dimensions. Accordingly, a non-distinguishing attribute is represented as a down-node in at least one dimension that is identified by the attributes distribution. Up-node polyarchies are also discovered when all the values of a down-node object are located in a substantially unique up-node object.

Attributes that are distributed across a substantially large set of objects have a locating domain property (e.g., a surname may be a locating domain property). Attributes with locating domain properties are used to narrow a search for particular ones of the data objects in the data polyarchy 122. Attributes that are distributed across a substantially small set of objects have a classifying domain property. Attributes with the classifying domain property are used to filter out unwanted objects from a search or navigation procedure.

Jump Gates

A sub-object entity reference 220 such as a GUID not only indicates whether a virtual object 210 (i.e., a respective object) has a relationship to a different object in the data polyarchy 122, but it also references the different object (i.e., via the different object's corresponding virtual object). Specifically, a sub-object reference uniquely identifies the different object of interest as a sub-object of the virtual object data field. The sub-object reference uniquely identifies the different object of interest across one or more data stores.

A virtual object 210 that references a sub-object (via a corresponding sub-object entity reference 220) is a "jump gate". A jump gate represents an elastic data relationship between a complex object and related sub-objects within the polyarchical data set 122. Inter-object data relationships in the data polyarchy are modeled with one or more simple objects 210 and/or complex objects 210. If an object has one or more sub-data relationships, such relationships are either represented as referenced sub-objects 220 in the object (or "surface entity"), or as separate objects 210 linked to another object 210 in some dimension.

To illustrate this, consider that an employee and the employee's dependents are people represented as objects in a directory store. The data store administrators may want to maintain fine-grained information about various aspects of each. To represent sub-world information (about the dependents) in the surface entity (the employee), one can use the following representation shown in TABLE 1.

TABLE 1
EXAMPLE OF STORING SUB-WORLD INFORMATION IN A SINGLE SURFACE ENTITY

```
<person type="employee" GlueID="13399">
  <name> John Doe </name>
  <age> 31 </age>
  <sex> male </sex>
  <dependents>
    <person type= "spouse">
      <name> Alice Doe </name>
      <age> 31 </age>
      <sex> female </sex>
    </person>
    <person type= "child">
      <name> Sigmund Doe </name>
      <age> 8 </age>
      <sex> male </sex>
    </person>
  </dependents>
  <occupation> forester </occupation>
</person>
```

To represent sub-world information about the dependents in totally distinct entities, Alice Doe and Sigmund Doe would be split off into separate entities, having their own Glue IDs (GUIDs 212), as illustrated, for example, in TABLE 2.

TABLE 2

EXAMPLE OF SEPARATE OBJECT/ENTITY REPRESENTATIONS

```
<person type= "spouse" GlueID= "24889">
  <relatedEmployee> 13399 </relatedEmployee>
  <name> Alice Doe </name>
  <age> 31 </age>
  <sex> female </sex>
</person>
<person type= "child" GlueID="24890">
  <relatedEmployee> 13399 </relatedEmployee>
  <name> Sigmund Doe </name>
  <age> 8 </age>
  <sex> male </sex>
</person>
```

Note that the "person" elements are identical whether they exist as sub elements in John's virtual entity or as root elements in their own independent virtual entities. In this context, John Doe's entity can be reduced as illustrated in TABLE 3.

TABLE 3

EXAMPLE OF A SINGLE ENTITY REPRESENTATION

```
<person type="employee" GlueID="13399">
  <name> John Doe </name>
  <age> 31 </age>
  <sex> male </sex>
  <occupation> forester </occupation>
</person>
```

The entity illustrated in TABLE 2 is related to John's dependents along the "dependents" dimension, where "relatedEmployee" is joined to Glue ID to "pass through the jump gate".

Between these two extremes, we can imagine representing John's node internally as illustrated in TABLE 4.

TABLE 4
EXAMPLE OF AN ENTITY REFERENCING ONE OR MORE OTHER ENTITIES

```
<person type="employee" GlueID="13399">
  <name> John Doe </name>
  <age> 31 </age>
  <sex> male </sex>
  <dependents>
    <person GlueID= "24889"/>
    <person GlueID= "24890"/>
  </dependents>
  <occupation> forester </occupation>
</person>
```

The entity of TABLE 4 could be returned to a client as is allowing the client to add to this information by expanding the related Glue IDs. Or a server such as a data polyarchy server 102 of Fig. 1 could itself de-reference the Glue IDs, returning the following amalgam (shown below in TABLE 5), and demonstrating the elasticity of the solution to the jump gate problem evident in traditional directory implementations.

TABLE 5

EXAMPLE OF DE-REFERNCED IDENTITY INFORMATION

```
<person type="employee" GlueID="13399">
  <name> John Doe </name>
  <age> 31 </age>
  <sex> male </sex>
  <dependents>
    <person type= "spouse" GlueID= "24889">
      <name> Alice Doe </name>
      <age> 31 </age>
      <sex> female </sex>
    </person>
    <person type= "child" GlueID= "24890">
      <name> Sigmund Doe </name>
      <age> 8 </age>
      <sex> male </sex>
    </person>
  </dependents>
  <occupation> archeologist </occupation>
</person>
```

In other words, a virtual object 210 can be modeled as either: (a) a simple object (often referred to as a "simple element") such as a character string, an integer, and so on, that does not reference any other element; or (b) a complex object (often referred to as a "complex element") that references one or more other simple elements or complex elements. In this manner, the polyarchical data set 122 provides for elastic inter-object data relationships that can be defined at any time with any one of a number of different relational representations.

Thus, in sharp contrast to traditional rigid directory implementations that have an intractable schema problem, wherein semantics of shape and naming in a directory must be agreed on in advance to solve the simplest design problem, no fundamental design decision is required when encountering an inter-object data relationship that is modeled as a jump gate. The shape and naming of the directory tree based on the polyarchical data set 122 is not affected by representing

various and elastic inter-object relationships even after a polyarchical data set has been designed. Moreover, an update/modification to a complex object may also result in corresponding updates to one or more related sub-objects that in turn may be represented in one or more different dimensions as compared to a particular dimension that represents the complex object.

Optimizing the Data Polyarchy Schema for De-referenced Operations

Two or more objects can be related to a third object for de-referenced dimensional group, or many-to-many object searching and navigation operations. For example, membership in a group is represented by a membership entity containing information about the relationship between a member and a group. A membership entity includes a *memberOf* data field to identify a group, and a *memberIs* data field to identify a group member. In this implementation such unique identification is accomplished by using respective GUIDs 212.

To determine if an entity is a member of a group, we search for a relationship entity where *memberIs* is the GUID of the entity, and *memberOf* is the GUID of the group. A membership dimension is defined as shown in TABLE 6.

TABLE 6

EXAMPLE OF A MEMBERSHIP DIMENSION IN SCHEMA

```
<dimension dereferenceElement="memberIs">
 <name> membership </name>
 <displayName lang="en"> Membership </displayName>
 <upnodeReferenceElement> memberOf </upnodeReferenceElement>
 <upnodeNamingElement> GlueID </upnodeNamingElement>
</dimension>
```

In this example, the group's GUID (represented in TABLE 6 as "GlueID") identifies the group as an upnode because the GUID is substantially unique, and the children are identified as membership entities with a *memberOf* element set to

the group's GUID. A conventional "down" navigation through the data set enumerates the membership entities--which may provide useful information about the nature of each individual membership (e.g. when a particular membership expires).

It is also possible to perform an "indirect" enumeration using the *memberIs* association to get information about the actual group members. To do so, issue a "down" enumeration on the group in the membership dimension with de-referencing set to *memberIs*. In this case, the membership entity's *memberIs* element is used to de-reference the actual entity belonging to the group. Thus, it is simple to construct an inverse dimension that list all groups belonged to by an entity. In this case, one may also either list the membership entities, or de-reference them to get information about the groups themselves.

Accordingly, no special schema design is required to represent a group's inverse polyarchies or other many-to-many inter-object relationships in the elements-of-interest schema 124.

An Exemplary Data Polyarchy Schema

Fig. 3 shows further aspects of an exemplary data polyarchy schema 124 of Fig. 1 to indicate how a client can manipulate the data polyarchy 122 in a meaningful manner. The data polyarchy schema is also referred to as an "elements-of-interest" schema. An element is an object attribute or data value. The elements-of-interest schema 124 includes a plurality of data fields 310 to limit a client 110 query on the data polyarchy. Such a query is communicated to the data polyarchy server 102 of Fig. 1. More specifically, such a query is

communicated to the polyarchy data management module 120 for processing. The query is limited to at least one subset of objects represented by the schema 124.

The elements 310 are not the objects themselves, but rather object representations (i.e., virtual objects 210 of Fig. 2) that indicate the relative scope of object data with respect to its distribution in the data polyarchy 122. As noted above, these virtual entities are stored as vectors or arrays of addresses in the schema.

Each different type of attribute 214 that an object 210 in the data polyarchy 122 could have is also identified in the schema as well as what kinds of indexes are to be used on the various attribute types.

The elements 310 (i.e., index types) are selected based on the relative distribution of the values assumed by an attribute within the data polyarchy 122. (The relative distribution of the values assumed by an attribute was discussed above in reference to Fig. 2). The elements 310 include at least one subset of the logical domain properties corresponding to all of the objects in the data polyarchy 122. (Logical domain properties are discussed above in reference to Fig. 2). The elements 310 represent attributes that have a substantially unique or "distinguishing" logical property index type, a locating, logical property index type, and/or a classifying logical property index type. Accordingly, the elements 310 include distinguishing elements 310-1, locating elements 310-2, and classifying elements 310-3.

A distinguishing element 310-1 (i.e., distinguishing index type) is a good candidate for a dimensional relationship between attributes in the data polyarchy 122 and is represented, for example, by a unique object (i.e., an object that has an attribute that is indexed by the distinguishing element) representing an up-node in

a dimension or hierarchy (e.g., a GUID, a location, an employee number, a cost center, and so on). The locating index type 310-3 or selecting index type is a good candidate for locating objects within the data polyarchy and is represented, for example, by the following attributes: a surname, a building name, a title, a room number, and/or the like. An attribute having a classifying index type such as an indication of gender (e.g., male or female) is a good candidate to filter objects in a search of objects in the data set because classifying objects are relatively small in number in the data polyarchy as compared to the relative distribution of objects with attributes that correspond to other index types.

The elements-of-interest schema 124 is highly customizable. For instance, a network administrator can assign natural language names such as names in English, French, Chinese, etc., to the elements, or objects in the elements-of-interest data set 124. Moreover, the administrator can designate sub-objects for storage as linked but discreet entities, as described in greater detail with respect to jump gates and TABLES 1 through 4. In this manner, objects in the polyarchical data set 122 of Figs. 1 and 2 that would not otherwise be immediately subordinate to a root object become eligible for promotion in the schema. This mechanism is used in conjunction with multiple dimensions (polyarchy) to produce elastic jump gates.

TABLE 7 shows an exemplary elements-of-interest schema 124 in an XML data format. Other data format representations besides XML representations (e.g., an extended version of XML, which has at least a subset or more of the features of XML) of elements 310 are contemplated. In this schema representation, boxed text (i.e., text boxed-in or surrounded with lines) and text preceded by a semi-

colon ";" represent corresponding comments. Generally comments of more than a single line are placed in a box.

TABLE 7

EXAMPLE ELEMENTS OF INTEREST SCHEMA

```
<WellKnownEntities GlueID="d7a5f1a9-6ba9-48a2-a464-660d82c24b5c">
        ; The "WellKnownEntities GlueID" tag is a unique schema ID.
        <ElementsOfInterest>            ; the beginning of the schema
            <element name="objectType">    ; name of the attribute
             <displayName lang="en" value="Object Type"/>
```

The "displayname lang" tag indicates a language ("lang") such as English ("en") and the corresponding value of the attribute in that language (e.g., cn's English display name is "Name"). As can be appreciated, the schema can be configured by a network administrator to indicate a number of different display names even for a single attribute (e.g., an English display name, French display name, Chinese display name, and so on).

```
    </element>
            <element name="GlueID" indexType="Distinguishing">
```

;Note that the "GlueID" element is identified as a "distinguishing" attribute. Other indexTypes include locating, or classifying index types. Each index type is based on the attributes relative distribution in the data polyarchy

```
             <displayName lang="en" value="Glue ID"/>
    </element>
            <element name="cn">
             <displayName lang="en" value="Name"/>
    </element>
            <element name="telephoneNumber">
             <displayName lang="en" value="Phone Number"/>
    </element>
            <element name="roomNumber">
             <displayName lang="en" value="Room Number"/>
    </element>
            <element name="uid">
             <displayName lang="en" value="E-mail Alias"/>
    </element>
            <element name="description">
             <displayName lang="en" value="Description"/>
    </element>
                                <element name="sn" indexType="selecting"
```

```
            <displayName lang="en" value="Surname"/>
    </element>
            <element name="givenName" indexType="locating"
startingSize="20000">
            <displayName lang="en" value="Given Name"/>
    </element>
            <element name="mail" indexType="distinguishing"
indexStartingSize="20000" indexGrowBy="20000">
            <displayName lang="en" value="E-mail Address"/>
    </element>
            <element name="buildingName" indexType="classifying">
            <displayName lang="en" value="Building Name"/>
    </element>
            <element name="title" indexType="classifying">
            <displayName lang="en" value="Title"/>
    </element>
            <element name="location" indexType="distinguishing">
            <displayName lang="en" value="Location"/>
    </element>
            <element name="locationUpnode"/>
            <element name="uniqueIdentifier" indexType="distinguishing">
            <displayName lang="en" value="Employee Number"/>
    </element>
            <element name="manager">
            <displayName lang="en" value="Manager"/>
    </element>
            <element name="costCenter" indexType="distinguishing">
            <displayName lang="en" value="Cost Center"/>
    </element>
            <element name="costCenterUpnode"/>
        </ElementsOfInterest>
        <Dimensions>
```

The "Dimensions" tag is a portion of the schema that identifies those objects in the data polyarchy 122; that represent a root node in a hierarchy of data relationships.

```
            <dimension>          ; indicates a dimension
                <name>costCenter</name>          ; name of the dimension

        <upnodeReferenceElement>costCenterUpnode</upnodeReferenceEleme
nt>
```

The "upnodeReferenceElement" tag represents —the element that contains the value that is present in the parent dimension naming element.

<dimensionNamingElement>costCenter</dimensionNaming Element>

The "dimensionNamingElement" tag names the objects in that dimension.

<view>

The "view" tag can indicate that objects above or below the dimension can be shown (e.g., siblings).

<displayName lang="en">Business Units</displayName>

<SearchType>nodeQuery</SearchType>

The "SearchType" tag indicates how the client should generate the query. If absent, the client will generate a query that returns a simple list. If "nodeQuery", the generated query will request a hierarchy result in the specified dimension. If "nodeConstraintQuery", the generated query will request a hierarchy constrained to only entities below the specified entity in the specified dimension. If "nodeExclusiveQuery", the generated query will request a hierarchy constrained to be below the first specified entity and NOT below the second and succeeding specified entities in the specified dimension.

<up>*</up>

The "up" tag the number of hierarchy levels (i.e., ancestors in the hierarchy) that are to be displayed in the dimension. In this case, the wild card "*" indicates that all levels in this dimension can be viewed.

<ElementsList>

These attributes indicate those that will be displayed at this level. This list is customizable.

<element>cn</element>
<element>uid</element>
<element>telephoneNumber</element>
<element>title</element>
<element>buildingName</element>
<element>roomNumber</element>
<element>description</element>
<element>companyCode</element>
<element>costCenter</element>

```
                    </ElementsList>                    </view>
            </dimension>
            <dimension>
                <name>Management</name>

        <upnodeReferenceElement>manager</upnodeReferenceElement>

        <dimensionNamingElement>uniqueIdentifier</dimensionNamingEleme
nt>
                <view>
                    <displayNamelang="en">Management</displayName>
                    <displayName lang="fr">Gestion</displayName>
                    <SearchType>nodeQuery</SearchType>
                    <up>*</up>
                    <ElementsList>
                        <element>cn</element>
                        <element>uid</element>
                        <element>telephoneNumber</element>
                        <element>title</element>
                        <element>buildingName</element>
                        <element>roomNumber</element>
                    </ElementsList>
                    <selected>true</selected>
```

The "selected" tag indicates to the client that this view is the default (selected) view in the client interface.

```
                </view>
                <view>
                    <displayName                    lang="en">Direct
Reports</displayName>
                    <SearchType>nodeQuery</SearchType>
                    <up>0</up>
```

The "<up>0</up>" tag is a dimensional direction indicator that indicates to the client that no hierarchy ancestors of the specified entity should be returned in the specified dimension.

```
                    <down>1</down>
```

The "<down>1</down>" tag is a dimensional direction indicator that indicates to the client that only the immediate descendants (one level of the hierarchy) of the specified entity should be returned in the specified dimension.

```
                    <ElementsList>
                        <element>cn</element>
```

```
                    <element>uid</element>
                    <element>telephoneNumber</element>
                    <element>title</element>
                    <element>buildingName</element>
                    <element>roomNumber</element>
                </ElementsList>
            </view>
            <view>
                <displayName lang="en">Related
People</displayName>
                <SearchType>nodeQuery</SearchType>
                <up>*</up>
                <down>1</down>
                <siblings>true</siblings>
```

The SIBLINGS='true' tag indicates that all objects with the same parent as the current object should be returned. The <up>, <down>, and <siblings> tags are not generated by any automatic analysis. They are carefully designed by a metaverse designer to enable a client to provide useful views to a user.

```
                <ElementsList>
                    <element>cn</element>
                    <element>uid</element>
                    <element>telephoneNumber</element>
                    <element>title</element>
                    <element>buildingName</element>
                    <element>roomNumber</element>
                </ElementsList>
            </view>
            <view>
                <displayName lang="en">Same Title (in
context)</displayName>
                <SearchType>nodeQuery</SearchType>
                <up>*</up>
                <SearchElement>title</SearchElement>
```

The "SearchElement" tag indicates to the client which element of interest should be queried on the selected entity. For example, if Jane Doe is selected and a "title" searchElement is specified, then the client will determine what Jane Doe's title is and do a search of all people with that title.

```
                    <ElementsList>
                        <element>cn</element>
                        <element>uid</element>
                        <element>telephoneNumber</element>
                        <element>title</element>
                        <element>buildingName</element>
                        <element>roomNumber</element>
                    </ElementsList>
                </view>
                <view>
                  <displayName lang="en">Same Title (list)</displayName>
                    <SearchType>nodeSearch</SearchType>
                    <SearchElement>title</SearchElement>
                    <ElementsLis>
                        <element>cn</element>
                        <element>uid</element>
                        <element>telephoneNumber</element>
                        <element>title</element>
                        <element>buildingName</element>
                        <element>roomNumber</element>
                    </ElementsLis>
                </view>
            </dimension>
            <dimension>
                <name>officeLocation</name>

        <upnodeReferenceElement>locationUpnode</upnodeReferenceElement
>

        <dimensionNamingElement>location</dimensionNamingElement>
                <view>
                    <displayName lang="en">Location of
Office</displayName>
                    <SearchType>nodeQuery</SearchType>
                    <up>*</up>
                    <ElementsLis>
                        <element>cn</element>
                        <element>uid</element>
                        <element>telephoneNumber</element>
                        <element>title</element>
                        <element>buildingName</element>
                        <element>roomNumber</element>
                        <element>description</element>
```

```
                    </ElementsLis>
                </view>
            </dimension>
        </Dimensions>
    <Inputs>
```

The "Inputs" tag indicates to the Polyarchy data manager where the source material is located and what element should be used as the anchor for that material.

```
    <Input name="base" path="input.xml" anchor="GlueID" />
    </Inputs>
</WellKnownEntities>
```

Exemplary Procedure to Dynamically Generate a Data Polyarchy

Fig. 4 illustrates an exemplary procedure 400 to generate multiple hierarchies of inter-object relationships based on the values of attributes of the objects. The data polyarchy 122 includes multiple objects. The procedure may be implemented in software as computer-executable instructions stored in a computer-readable medium such that when executed by a processor that is operatively coupled to the medium, the instructions perform the operations described in the blocks of Fig. 4.

At block 410, the data polyarchy server 102 of Fig. 1 receives data from any number of data sources such as from a conventional directory service based on X-500 and LDAP, metadirectory service, a database, and so on. The data is received in any one of a number of different data formats such as the XML data format. The server 102 communicates the received data to the data polyarchy management module 120 of Fig. 1.

At block 412, responsive to receiving the data (block 410), the data polyarchy management module 120 generates or updates the data polyarchy 122 to reflect any inter-object relationships (e.g., mono-directional and/or bi-directional

relationships) between the received data and the data (if any) already in the polyarchy 122. As already discussed, these inter-object relationships are determined based on the attributes of the received data with respect to the attributes of the other objects in the polyarchy. Specifically, to generate, configure, or update the data polyarchy, the management module analyzes the relative distributions of the attributes of the objects in the data polyarchy to determine which of zero, one, or more dimensions within which each object participates in inter-object relationships with other objects in the polyarchy.

These operations 412 are automatic or dynamic responsive to receipt of the data (block 410) and do not require any intervention of any human operators such as network administrators. Because inter-object relationships in the data polyarchy 122 are determined and expressed based on the values of attributes of the objects in the polyarchy, these inter-object data relationships can be elastic—meaning that they can change over time. As values of attributes change, the inter-object relationships based in the new values are dynamically or automatically represented in the polyarchy by the management module 120 upon receipt. These operations 412 are performed independent of a-priori knowledge of data relationships between respective ones of the data objects in the data polyarchy. Additionally, because inter-object relationships in the data polyarchy are determined and expressed based on the values of attributes of the objects in the polyarchy, these relationships are determined and expressed completely independent of a distinguished name of an object.

At block 414, the data polyarchy management module 120 of Fig. 1 generates, configures, or updates the elements-of-interest schema 124 (e.g., see Figs. 1 and 3) to indicate how the data polyarchy 122 can be manipulated,

presented, and navigated in a meaningful manner. Specifically, as discussed above in reference to Figs. 2 and 3, and Table 7, the schema indicates the elements, or attributes in the data polyarchy along with any corresponding distinguishing, locating, or classifying characteristics of each attribute. The schema also indicates the dimensions in the polyarchy along with each attribute or element of interest contained by objects in the dimension.

An exemplary set of polyarchical query language (PQL) commands (based on the elements-of-interest schema 124) used by a browser to search, navigate, or display portions of the polyarchical data set 122 are described in greater detail below in reference to Figs. 6 through 12. An exemplary procedure to use the elements-of-interest schema 124 to formulate PQL requests and responses is described in greater detail below in reference to Fig. 13.

Exemplary Polyarchical Query Language Request

Fig. 5 shows an exemplary polyarchical query language (PQL) query used by a client 110 to request a data polyarchy server 102 to return information (a PQL response) corresponding to information in the data polyarchy. . Responsive to receiving such a query, the data polyarchy management module 120 identifies and retrieves a set of information corresponding to objects in the polyarchy

Queries 500 and corresponding server 102 responses are implemented using a text markup language such as XML. In this configuration, the queries and server responses are packaged in a Simple Object Access Protocol (SOAP) and posted over the network 104 of Fig. 1 using the Hypertext Transfer Protocol (HTTP). SOAP and HTTP are communication protocols that are well known to those skilled in the art of network communication protocols.

The message 500 includes a schema ID 502 and one or more object transformation parameters 510 (hereinafter, a parameter is also referred to as a data field) for specifying one or more attributes 214 of Fig. 2. The schema ID is used to identify a particular elements-of-interest schema 124. It can be appreciated that this data field is optional if there is a default schema or only one schema. The attributes 510 correspond to the virtual objects 210 of the data polyarchy 122. (The attribute(s) include distinguishing attributes, locating attributes, or classifying attributes, each of which is discussed in greater detail above with respect to logical domain properties of Fig. 2).

A parameter 510, or data field is classified according to its type, which is selected from types that include a specific element of interest type 510-1; an elements-of-interest modifier to limit a response 510-2; a Boolean modifier 510-3; a dimension indicator 510-4; and/or a dimension information modifier 510-5. The number and types of data fields that are represented in the message 500 are based on the message's design, or purpose.

Fig. 6 shows a user interface (UI) 600 displaying an exemplary PQL query 500 message and a corresponding exemplary PQL response 620. Specifically, the PQL query includes a modifier parameter 510-1 based on a data polyarchy schema 124 to specify a particular attribute 510 with which to perform a search of the data polyarchy 122. The UI includes a first area 610 to type in a PQL message 500, a second area 612 to show the PQL message packaged in a SOAP envelope 618 and posted over HTTP, and a third area 614 to show the data polyarchy management module 120 PQL response 620. Although the PQL response is shown as being returned in a SOAP envelope, the response can be returned in a variety of other data packaging formats.

In this example, the specific element of interest parameter 510-1 specifies a surname attribute "Doe". The PQL response 620 returned at least two objects and corresponding elements of interest. A respective Glue ID identifies each respective object, which is a distinguishing element. The first object pertains to "John Doe". The second object pertains to "Jim Doe". Each object was returned with a number of elements-of-interest such as a room number, a user id ("uid"), a surname ("sn"), a given name, a building name, a title, an indication of a related dimension ("locationUpnode"), the entities manager ("manager"), cost center id, and the like.

If the specific element of interest specified an absolutely unique distinguishing attribute such as a GUID that corresponds to a particular object stored in a data polyarchy 122, the server 102 will return all of the information stored in the data polyarchy 122 with respect to the particular object.

Fig. 7 shows user interface 600 displaying an exemplary PQL query with an elements-of-interest modifier data field 510-2 that specifies a limiting attribute with which to modify a result of a search. The limiting attribute corresponds to a set of objects represented by a polyarchical data schema 124. The elements-of-interest modifier data field indicates to a server that a response to a search operation is limited to presenting any identified data polyarchy 122 objects with respect to the limiting attribute.

In this example, the limiting attributes 510-2 are a common name ("cn") attribute and a unique identifier attribute. Thus, the various person objects 620 returned by the server indicate only those limiting attributes.

Fig. 8 illustrates a user interface for an exemplary PQL query 500 that includes a Boolean modifier parameter 510-3 to perform a mathematical operation

with respect to polyarchies of data relationships. A Boolean modifier is used to perform a filtering operation ("and"), a union operation ("or"), or an exclusion operation ("not") on one or more hierarchies of data relationships based on variable. The variable includes an object represented by the data polyarchy schema 124 of Figs. 1 and 3, and Table 7, a hierarchy of objects represented by the schema, or polyarchies of objects represented by the schema, and so on.

For example, the "and" Boolean modifier 510-3 is used to filter the results of two data store searches based on specific elements-of-interest data fields 510-1. A first specific elements-of-interest data field specifies a surname ("sn") attribute with a value of "Smith". A second specific elements-of-interest data field specifies a "title" attribute with a value of "vice president". Thus, the Boolean modifier is used to narrow, or filter the results based on the respective search results. The result is a single object in the PQL response 620 that corresponds to vice president John Smith. If there were more than one set of entity information stored in a directory that matched this query 500, then each of the entities would be presented in the result.

Fig. 9 shows a user interface 600 that in turn illustrates an exemplary PQL query 500 that includes a dimension information indicator data field 510-4 for specifying a dimension within which to present a response that corresponds to a search operation for an object stored in a data store. In this example, the "under" parameter 510-4 (or "clause") is combined with a filter 510-3 ("<and>") to find "architects" under John Smith, which as indicated has a corresponding "unique identifier" of "1234567898". (See, also John Smith's unique identifier of Fig. 8). Information corresponding to the architects under John Smith is presented in the PQL response 620 from the server 102. (Note how an elements-of-interest data

field 510-2 was used to limit the number of elements presented in the results of the search).

Fig. 10 shows user interface 600 for illustrating a PQL query 500 with a dimension information modifier data field 510-5. The dimension information modifier specifies a particular direction and a particular depth to present a data relationship between a complex object in a polyarchical schema and one or more different represented objects. The direction indicates whether the one or more (all objects with the use of a wildcard indication such as "*") different objects are sub-objects of the complex object. The dimension information modifier can also specify SIBLINGS='true' to indicate that all objects with the same parent as the current object should be returned.

In this example, the dimension information modifier 510-5 is used to retrieve information 620 corresponding to a first level of subordinates 1010 from a data store. This is a jump gate because John Smith's subordinates 1010 are presented as aspects of John Smith's object definition 620.

Fig. 11 is a block diagram of a user interface 600 showing use of a filter parameter in a PQL query 500 with respect to a particular attribute and a subsequent intersection between two polyarchies of data relationships. In this example, two dimensions 510-4 (e.g., a "management" dimension and an "office location" dimension) are intersected and filtered 510-3 based on a "title" attribute of "architect". The search results 620 show the particular objects in the data store that match that query.

Fig. 12 shows the user interface 600 for illustrating a PQL 500 that specifies a filter ("and") 510-3 and a union ("or") 510-3 between two polyarchies 510-4 of data relationships. In this example, the filter and the union

are Boolean modifiers. The union attribute is applied to the "management" dimension and the "office location" dimension. The filter specifies a "title" attribute of "architect", which is then applied to the union of the two hierarchies. The search results 620 show the particular objects in the data store that match that query.

Exemplary Procedure to Manage a Polyarchical Data Set

Fig. 13 shows an exemplary procedure 1300 to manage data in a data polyarchy 122. At block 1310, the polyarchical data management module 120 communicates an elements-of-interest schema 124 to a client 110. The elements-of-interest schema 124 indicates to the client how objects in the data polyarchy can be accessed, manipulated, and presented by the client in a meaningful manner.

At block 1312, the polyarchical data management module 120 receives a PQL message 500 that is based on the communicated data polyarchy schema (block 1310). The request not only identifies one or more attributes of interest but also identifies the data relationships of interest. The request corresponds to a data object of the data objects in the polyarchical data set 122 of Fig. 1.

The received PQL message 500 may correspond to one or more operations including any combination of: (a) an operation to find a default search object of the data objects; (b) an operation to locate an object of the data objects that corresponds to a particular name; (c) an operation to obtain a default hierarchy of data relationships that correspond to a particular object of the data objects; (d) an operation to obtain a particular hierarchy of data relationships that correspond to a particular object of the data objects; (e) an operation to identify at least one subset of a plurality of hierarchies of data relationships that correspond to a particular

object of the data objects; (f) an operation to obtain multiple hierarchies of data relationships that correspond to a particular object of the data objects; and so on.

At block 1314, the data polyarchy management module 120 determines a physical access strategy (e.g., a simple scan, a recursive scan, and so on) to identify data corresponding to the request from the data polyarchy 122. This determination is based on the request (block 1312), which in turn is based on the schema 124 that was communicated to the client 110 (block 1310). As already noted, the schema provides the client not only with information that corresponds to the possible contents of the data polyarchy, but also with includes information describing the possible polyarchies of data relationships that may pertain to any one object of interest (e.g., see the "<Dimension>" indicators shown in Table 7).

For instance, consider that if a client request (i.e., a PQL message 500) is designed to filter out all elements-of-interest that pertain to an object with the exception of an absolutely unique distinguishing attribute (e.g., a GUID and a common name that corresponds to the GUID), a simple scan of the data polyarchy 122 is an efficient technique to search for information regarding the distinguishing object of interest.

The request 500, however, may also indicate that a number of sub-objects should be presented with respect to a complex object (i.e., a jump gate) and then the results are to be subsequently modified by a union of a dimension of information that corresponds to the complex object that is orthogonal to one or more of the sub-objects. In this case, a recursive scan of the data polyarchy 122 is an efficient technique to search for information regarding the objects and inter-object relationships of interest.

In this manner, a PQL request message 500 identifying attributes and data relationships of interest also provides an optimized physical access strategy to search the data polyarchy 122 for such attributes and data relationships.

At block 1316, the data polyarchy management module 120 accesses the data from the polyarchy based on the determined physical access strategy (block 1314). The accessed data may take a number of different forms. For instance, the accessed data may be independent of any inter-object relationship between the data object and any other object in the polyarchy. Additionally, the accessed object(s) may participate in one or more hierarchies of inter-object relationships with one or more different data objects in the polyarchy. In this case, the accessed object(s) and the one or more different objects comprise a similar attribute. As discussed above, these inter-object relationships may be orthogonal with respect to one another in one or more dimensions.

At block 1318, the polyarchical data management module 120 transforms the accessed data for issuing to the client 110. Specifically, accessed data is transformed based on the requirements of the specific PQL message 500 that was used to request the data (block 1312). For instance, if the message indicates an object with respect to a particular dimension, the implicit and explicit inter-object relationships of the accessed data are assembled into a hierarchy based on the particular dimension.

For example, an accessed data object represents a jump gate when the accessed data includes a complex object of the data objects that is related to one or more sub-objects of the data objects. In this case the complex object is transformed or represented as an independent surface entity. Each of the one or more sub-objects is described as a respective separate entity in a manner that is

independent of the surface entity. The one or more sub-objects are then transformed or referenced in the surface entity to indicate a relationship between the complex object and the one or more sub-objects. The referencing is independent of any object naming or hierarchical data relationship between the complex object and the one or more sub-objects.

In another example, accessed data includes a first object of the data objects in the polyarchy that is related to one or more sub-objects. The first object is transformed or represented as an independent surface entity. Each of the one or more sub-objects is described as respective separate entities in a manner that is independent of the surface entity. Then, a respective link is included in each of the one or more sub-objects to reference the first object. In this manner, as in the previous example, the data is transformed to express the relationship of interest as indicated in the corresponding PQL message 500.

At block 1320, data polyarchy management module 120 issues, or communicates the transformed data (block 1318) to the client.

Exemplary Computing Environment

Fig. 14 illustrates an example of a suitable computing environment 1400 on which an exemplary data polyarchy server 102 of Fig. 1 may be implemented. The exemplary computing environment is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of an exemplary data polyarchy server 102, a server 106, or a client 110. Neither should the computing environment 1400 be interpreted as having any dependency or requirement relating to any one or

combination of components illustrated in the exemplary computing environment 1400.

The computer 1402 is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with an exemplary computer 1402 include, but are not limited to, personal computers, server computers, thin clients, thick clients, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

An exemplary computer 1402 may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, and so on, that performs particular tasks or implements particular abstract data types. An exemplary computer1402 may be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

As shown in Fig. 14, the computing environment 1400 includes a general-purpose computing device in the form of a computer 1402. The components of computer 1402 may include, by are not limited to, one or more processors or processing units 1412, a system memory 1414, and a bus 1416 that couples

various system components including the system memory 1414 to the processor 1412.

Bus 1416 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnects (PCI) bus also known as Mezzanine bus.

Server 1402 typically includes a variety of computer readable media. Such media may be any available media that is accessible by computer 1402, and it includes both volatile and non-volatile media, removable and non-removable media.

In Fig. 14, the system memory 1414 includes computer readable media in the form of volatile memory, such as random access memory (RAM) 1420, and/or non-volatile memory, such as read only memory (ROM) 1418. A basic input/output system (BIOS) 1422, containing the basic routines that help to transfer information between elements within computer 1402, such as during start-up, is stored in ROM 1418. RAM 1420 typically contains data and/or program modules that are immediately accessible to and/or presently be operated on by processor 1412.

Computer 1402 may further include other removable/non-removable, volatile/non-volatile computer storage media. By way of example only, Fig. 14 illustrates a hard disk drive 1424 for reading from and writing to a non-removable,

non-volatile magnetic media (not shown and typically called a "hard drive"), a magnetic disk drive 1426 for reading from and writing to a removable, non-volatile magnetic disk 1428 (e.g., a "floppy disk"), and an optical disk drive 1430 for reading from or writing to a removable, non-volatile optical disk 1432 such as a CD-ROM, DVD-ROM or other optical media. The hard disk drive 1424, magnetic disk drive 1426, and optical disk drive 1430 are each connected to bus 1416 by one or more interfaces 1434.

The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules, and other data for computer 1402. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 1428 and a removable optical disk 1432, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, random access memories (RAMs), read only memories (ROM), and the like, may also be used in the exemplary operating environment.

A number of program modules 1440 may be stored on the hard disk, magnetic disk 1428, optical disk 1432, ROM 1418, or RAM 1420, including, by way of example, and not limitation, an operating system 1438, one or more application programs 1440, other program modules 1442, and program data 1444.

Each of such operating system 1438, one or more application programs 1440 (e.g., a polyarchy data management module 120), other program modules 1442, and program data 1444 (e.g., the data polyarchy 122 and the elements-of-interest schema 124)—or some combination thereof, may include an

implementation of an exemplary data polyarchy server 102 of Fig. 1. Specifically, each may include an implementation of a data polyarchy server 102 to:

(a) dynamically generate, manage, and update a data polyarchy 122 based on attribute values of the objects;

(b) analyze the data polyarchy based on relative distribution of attributes to generate an elements of interest schema indicating how objects in the data polyarchy can be meaningfully presented and manipulated within various inter-object relationships;

(c) communicate the elements-of-interest schema 124 to a client 110;

(d) responsive to receiving a query (e.g., a PQL message 500) based on the schema, determine a physical access strategy to access the requested data from a polyarchical data set 122;

(e) access and transform the data based on the query request; and,

(f) issue the transformed data to the client as a response.

A user may enter commands and information into computer 1402 through optional input devices such as keyboard 1446 and pointing device 1448 (such as a "mouse"). Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, serial port, scanner, or the like. These and other input devices are connected to the processing unit 1412 through a user input interface 1450 that is coupled to bus 1416, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB).

An optional monitor 1452 or other type of display device is also connected to bus 1416 via an interface, such as a video adapter 1454. In addition to the monitor, personal computers typically include other peripheral output devices (not

shown), such as speakers and printers, which may be connected through output peripheral interface 1455.

Computer 1402 may operate in a networked environment using logical connections to one or more remote computers, such as a remote server/computer 1462 (e.g., data servers 106). Remote computer 1462 may include many or all of the elements and features described herein relative to computer 1402.

Logical connections shown in Fig. 14 are a local area network (LAN) 1457 and a general wide area network (WAN) 1459. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet. When used in a LAN networking environment, the computer 1402 is connected to LAN 1457 via network interface or adapter 1466. When used in a WAN networking environment, the computer typically includes a modem 1458 or other means for establishing communications over the WAN 1459. The modem, which may be internal or external, may be connected to the system bus 1416 via the user input interface 1450 or other appropriate mechanism.

Depicted in Fig. 14, is a specific implementation of a WAN via the Internet. Computer 1402 typically includes a modem 1458 or other means for establishing communications over the Internet 1460. Modem, which may be internal or external, is connected to bus 1416 via interface 1450.

In a networked environment, program modules depicted relative to the personal computer 1402, or portions thereof, may be stored in a remote memory storage device. By way of example, and not limitation, Fig. 14 illustrates remote application programs 1469 as residing on a memory device of remote computer 1462. It will be appreciated that the network connections shown and described are

exemplary and other means of establishing a communications link between the computers may be used.

Computer Readable Media

An implementation of an exemplary computer 102 may be stored on or transmitted across some form of computer readable media. Computer readable media can be any available media that can be accessed by a computer. By way of example, and not limitation, computer readable media may comprise "computer storage media" and "communications media."

"Computer storage media" include volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by a computer.

"Communication media" typically embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as carrier wave or other transport mechanism. Communication media also includes any information delivery media.

The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes

wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media. Combinations of any of the above are also included within the scope of computer readable media.

Conclusion

The described arrangements and procedures replace traditional notions of distinguished names that represent inter-object relationships within a single static hierarchy. More specifically, the described arrangements and procedures replace these traditional notions with dynamically generated graphs of inter-object connections in multiple dimensions of data relationships based on attributes of the objects. In this manner, complex real-world objects are represented with respect to the particular objects themselves, with respect to any set of decomposed sub-entities, or sub-objects that are related to the particular objects. These inter-object relationships are managed and navigated using a data polyarchy schema 124 that has been generated to access elements of interest in the data polyarchy 122.

Although the described subject matter to generate and manage polyarchies of data relationships has been described in language specific to structural features and/or methodological operations, it is to be understood that the subject defined in the appended claims is not necessarily limited to the specific features or operations described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed present invention.

4. Brief Description of the Drawings

Fig. 1 shows an exemplary system for dynamically generating and managing multiple hierarchies of inter-object relationships based on the values of attributes of the objects.

Fig. 2 illustrates an exemplary polyarchy data structure to represent multiple hierarchies of dynamically generated inter-object relationships that are based on the values of attributes of the objects.

Fig. 3 shows an exemplary schema data structure to indicate how a data polyarchy of Fig. 2 can be created, accessed, and manipulated in a meaningful manner.

Fig. 4 shows an exemplary procedure to generate multiple hierarchies of inter-object relationships based on the values of attributes of the objects.

Fig. 5 shows an exemplary polyarchical query language (PQL) request used by a client to request a server to return information (a PQL response) from a data polyarchy.

Fig. 6 shows a user interface (UI) displaying an exemplary PQL query and a corresponding exemplary PQL response. Specifically, the PQL query includes a modifier parameter based on a data polyarchy schema to specify a particular attribute with which to perform a search of a polyarchical data set.

Fig. 7 is a block diagram of a UI displaying an exemplary PQL query and a corresponding exemplary PQL response. Specifically, the PQL query includes a modifier parameter based on an elements-of-interest schema; the parameter specifies a limiting attribute with which to modify a result of a search.

Fig. 8 is a block diagram of a UI displaying an exemplary PQL query and a corresponding exemplary PQL response. Specifically, the PQL query includes a Boolean modifier parameter to perform a mathematical operation with respect to polyarchies of data relationships.

Fig. 9 is a block diagram of a UI showing an exemplary PQL query and a corresponding exemplary PQL response. Specifically, the PQL query includes a

dimension information indicator parameter for specifying a dimension within which to view an object stored in a data store.

Fig. 10 is a UI showing an exemplary PQL query and a corresponding exemplary PQL response. Specifically, the PQL query includes a dimension information modifier parameter, which specifies a particular hierarchical direction and a particular hierarchical depth for a server process to present a data relationship between a complex object of a polyarchical data set and one or more other objects.

Fig. 11 is a UI showing use of a locating element in an exemplary PQL query with respect to a particular attribute and a subsequent intersection between two corresponding polyarchies of data relationships to form an exemplary PQL response.

Fig. 12 is UI showing an exemplary PQL query and a corresponding exemplary PQL response. Specifically, the PQL query illustrates use of filter and union parameters with respect to two polyarchies of data relationships.

Fig. 13 illustrates aspects of an exemplary procedure to manage data (e.g., to access, present, provide, and/or manipulate objects, etc.) in a data polyarchy (i.e., multiple hierarchies of dynamically generated inter-object relationships that are based on the values of attributes of the objects).

Fig. 14 shows aspects of an exemplary operating environment for managing a data polyarchy.

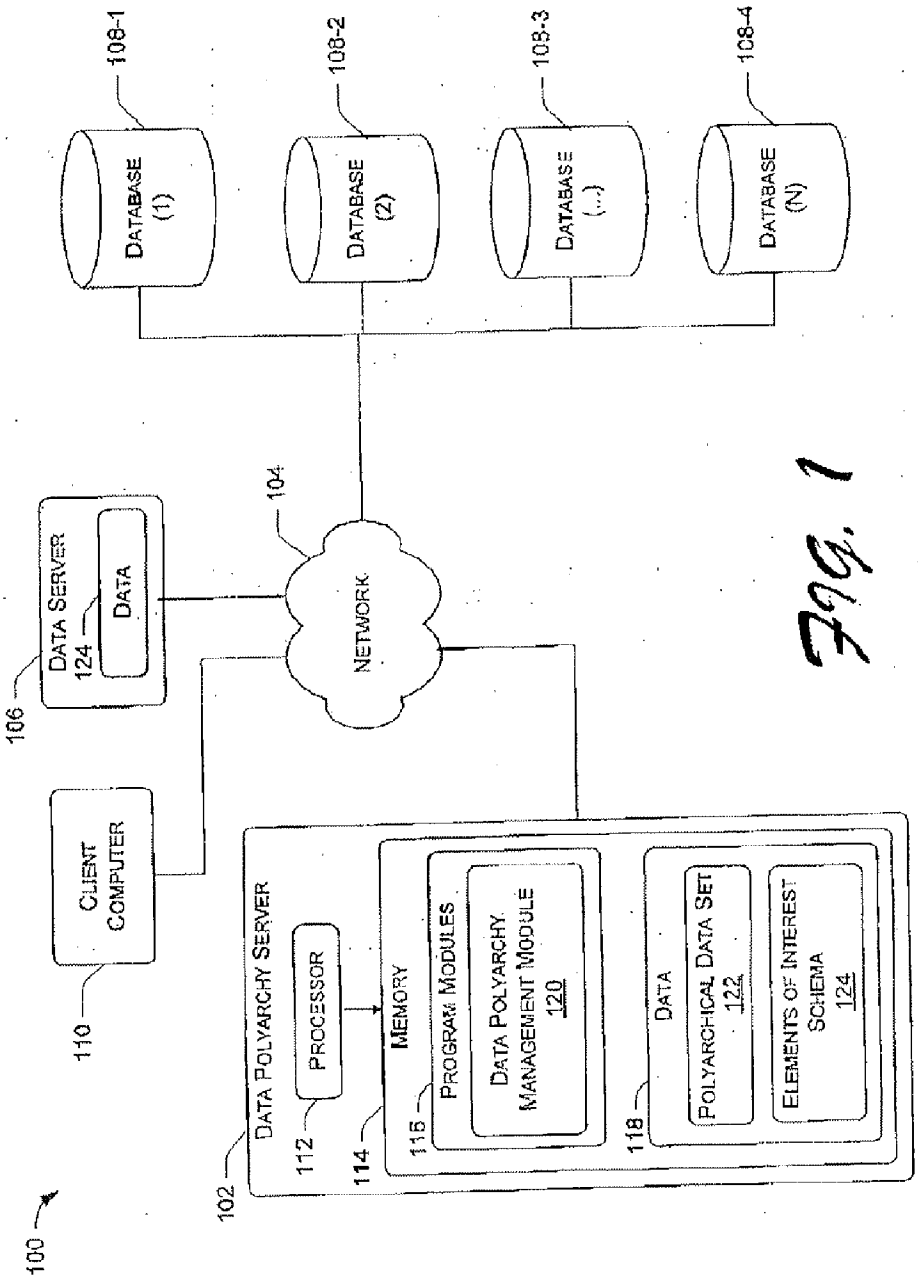

100
102
DATA POLYARCHY SERVER
PROCESSOR
112
MEMORY
114
PROGRAM MODULES
116
DATA POLYARCHY MANAGEMENT MODULE 120
DATA
118
POLYARCHICAL DATA SET 122
ELEMENTS OF INTEREST SCHEMA 124
CLIENT COMPUTER
110
DATA SERVER
124
DATA
106
NETWORK
104
DATABASE (1)
108-1
DATABASE (2)
108-2
DATABASE (...)
108-3
DATABASE (N)
108-4

Fig. 1

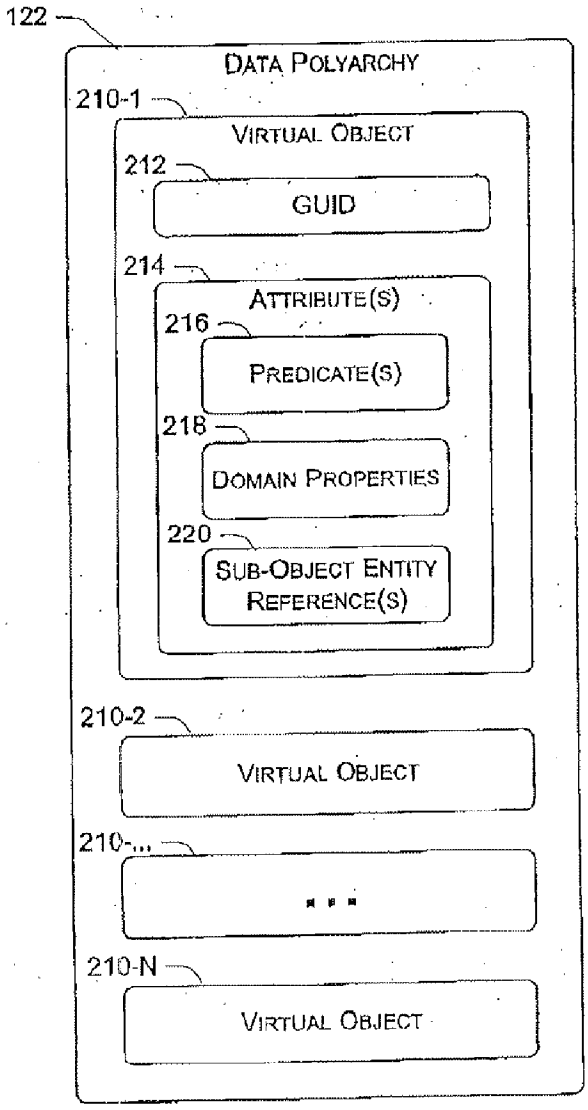

122
DATA POLYARCHY
210-1
VIRTUAL OBJECT
212
GUID
214
ATTRIBUTE(S)
216
PREDICATE(S)
218
DOMAIN PROPERTIES
220
SUB-OBJECT ENTITY REFERENCE(S)
210-2
VIRTUAL OBJECT
210-...
...
210-N
VIRTUAL OBJECT

FIG. 2

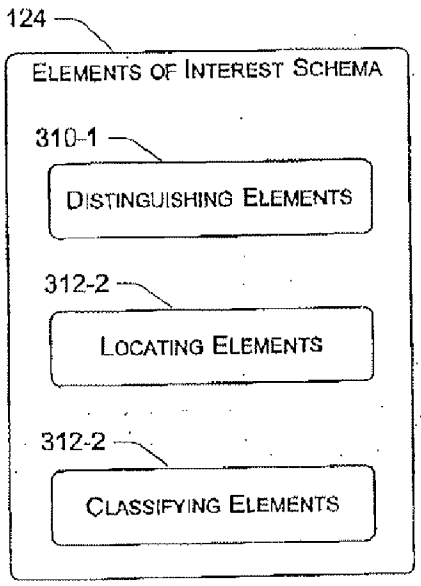

124
ELEMENTS OF INTEREST SCHEMA
310-1
DISTINGUISHING ELEMENTS
312-2
LOCATING ELEMENTS
312-2
CLASSIFYING ELEMENTS

FIG. 3

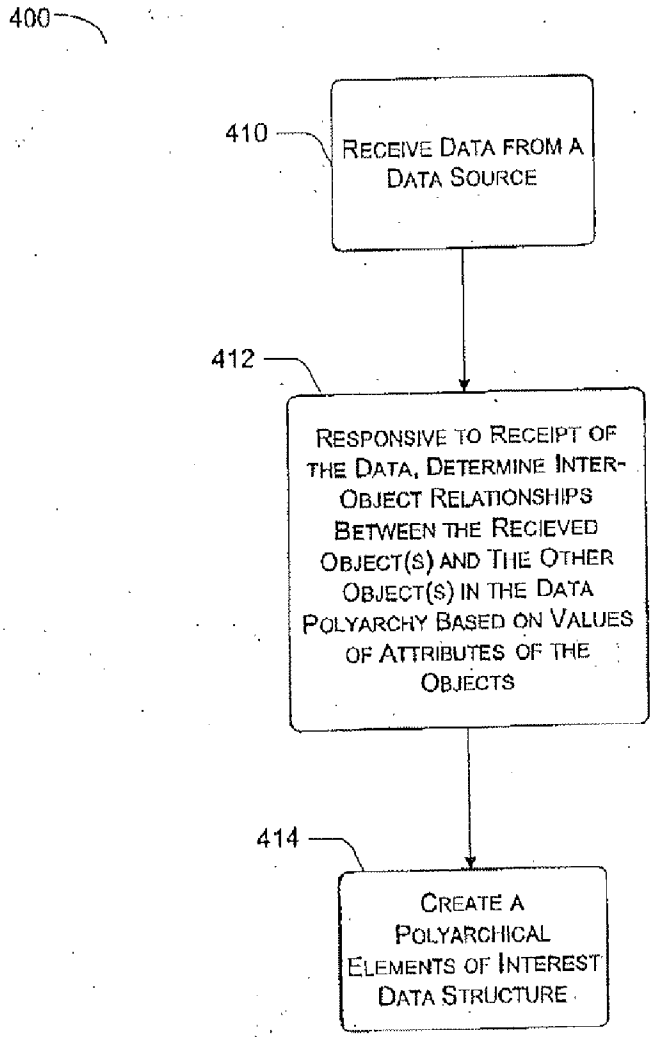

400
410
RECEIVE DATA FROM A DATA SOURCE
412
RESPONSIVE TO RECEIPT OF THE DATA, DETERMINE INTER-OBJECT RELATIONSHIPS BETWEEN THE RECIEVED OBJECT(S) AND THE OTHER OBJECT(S) IN THE DATA POLYARCHY BASED ON VALUES OF ATTRIBUTES OF THE OBJECTS
414
CREATE A POLYARCHICAL ELEMENTS OF INTEREST DATA STRUCTURE

Fig. 4

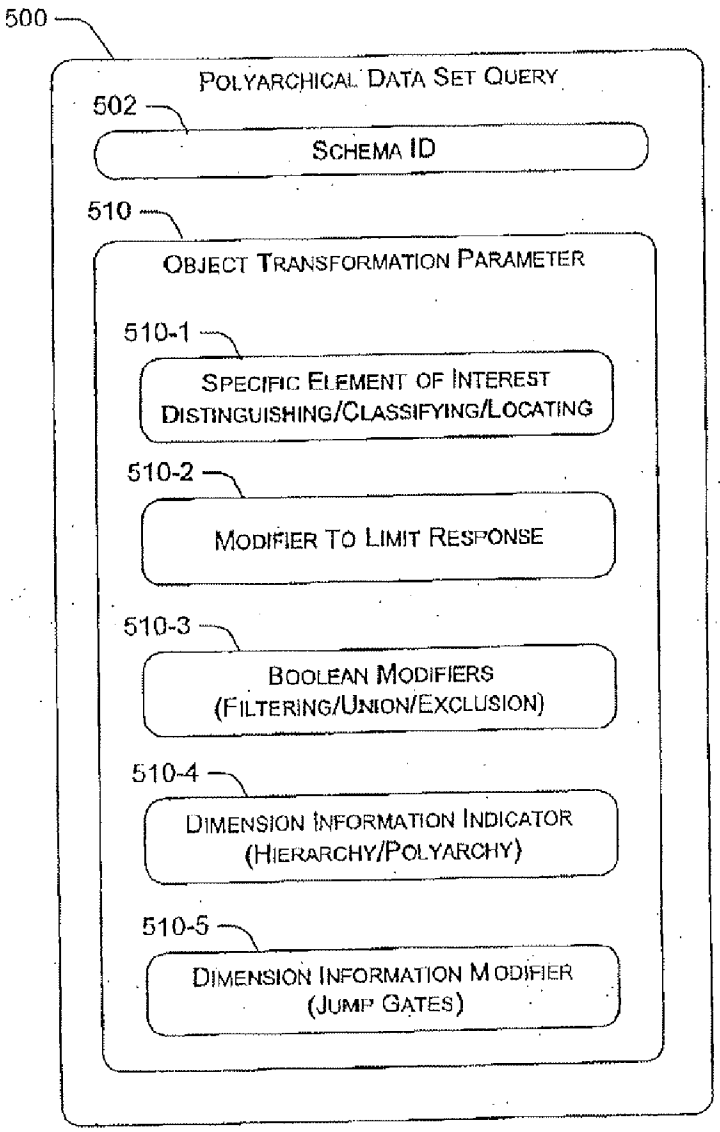

500
POLYARCHICAL DATA SET QUERY
502
SCHEMA ID
510
OBJECT TRANSFORMATION PARAMETER
510-1
SPECIFIC ELEMENT OF INTEREST DISTINGUISHING/CLASSIFYING/LOCATING
510-2
MODIFIER TO LIMIT RESPONSE
510-3
BOOLEAN MODIFIERS (FILTERING/UNION/EXCLUSION)
510-4
DIMENSION INFORMATION INDICATOR (HIERARCHY/POLYARCHY)
510-5
DIMENSION INFORMATION MODIFIER (JUMP GATES)

Fig. 5

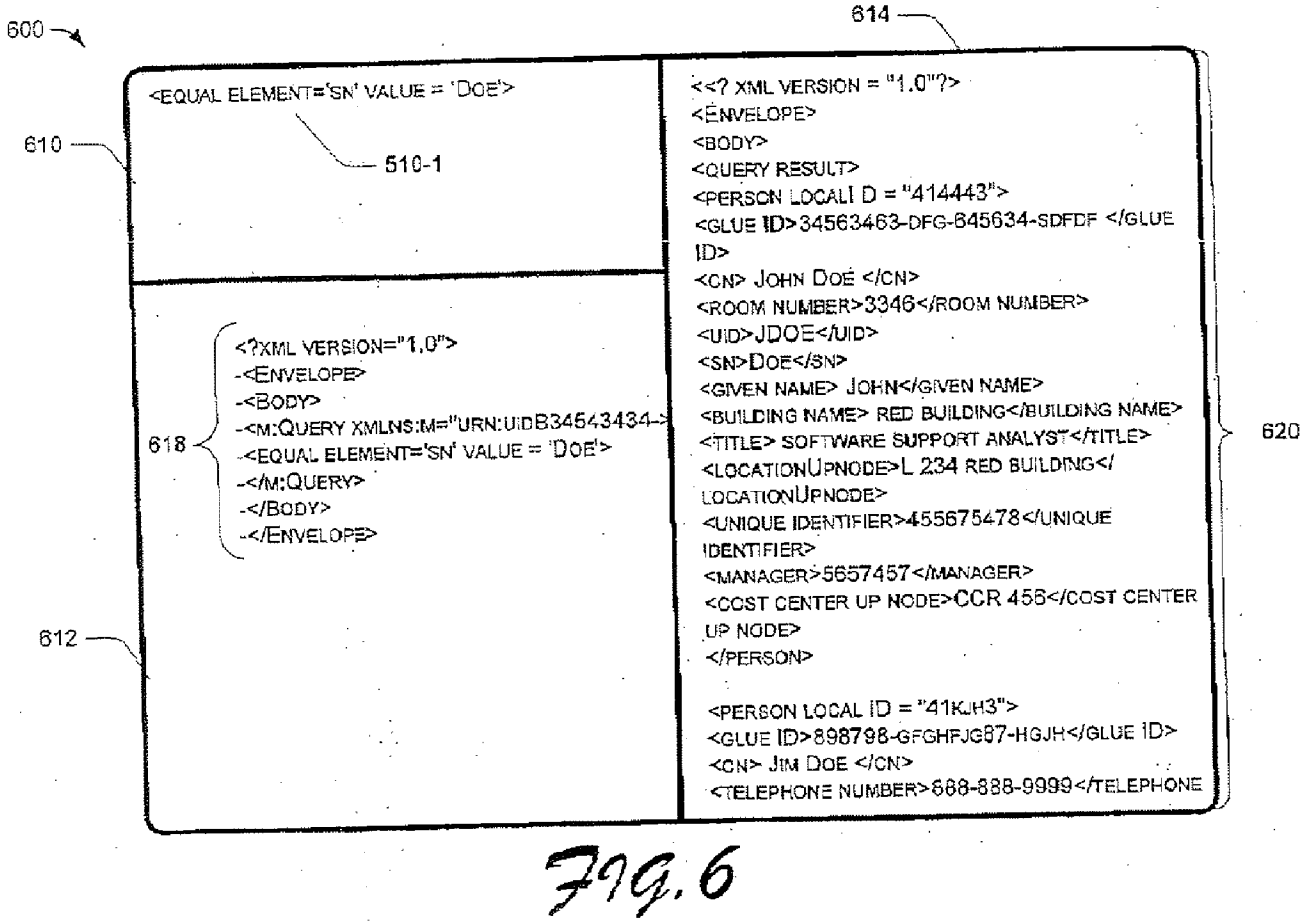


FIG. 6

(115) 特開2002-297433

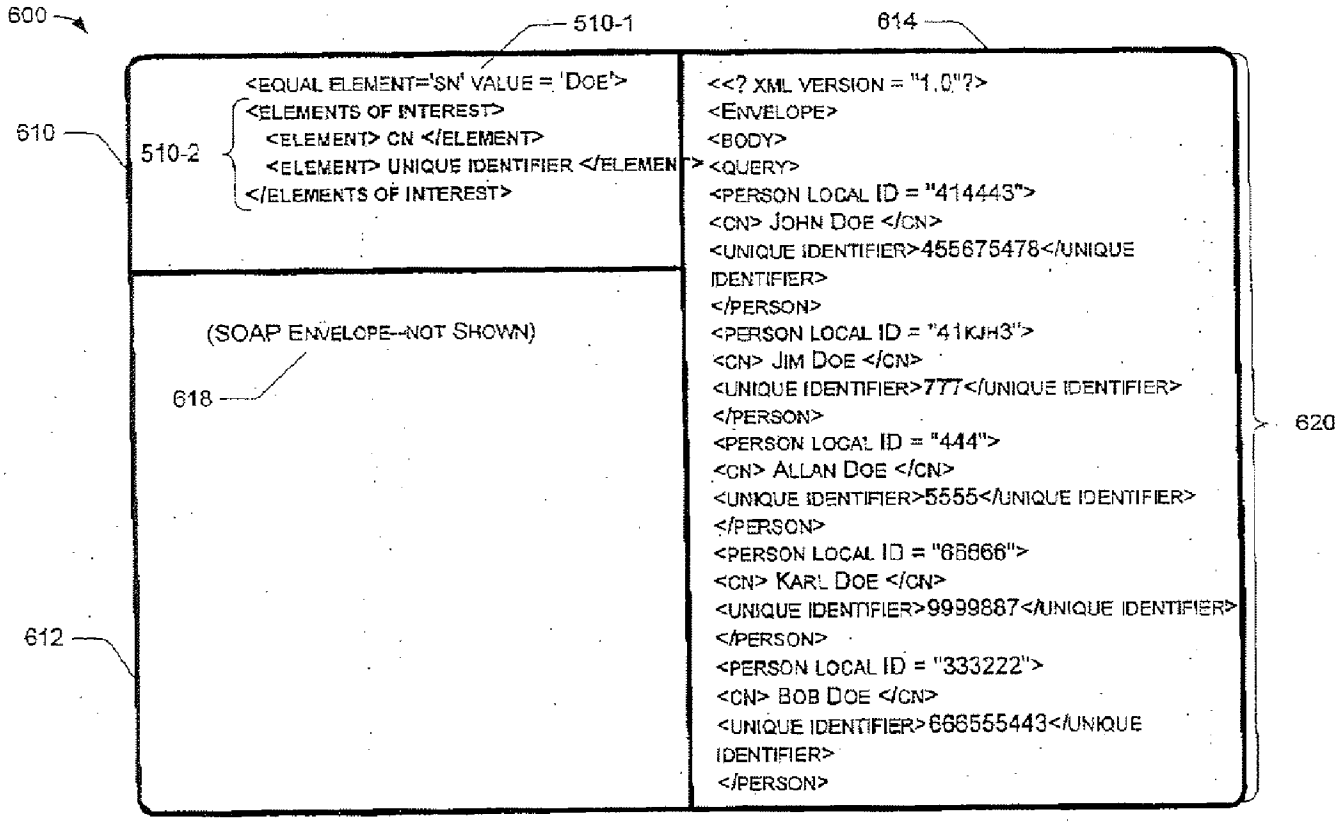


Fig. 7

(116)
特開2002-297433

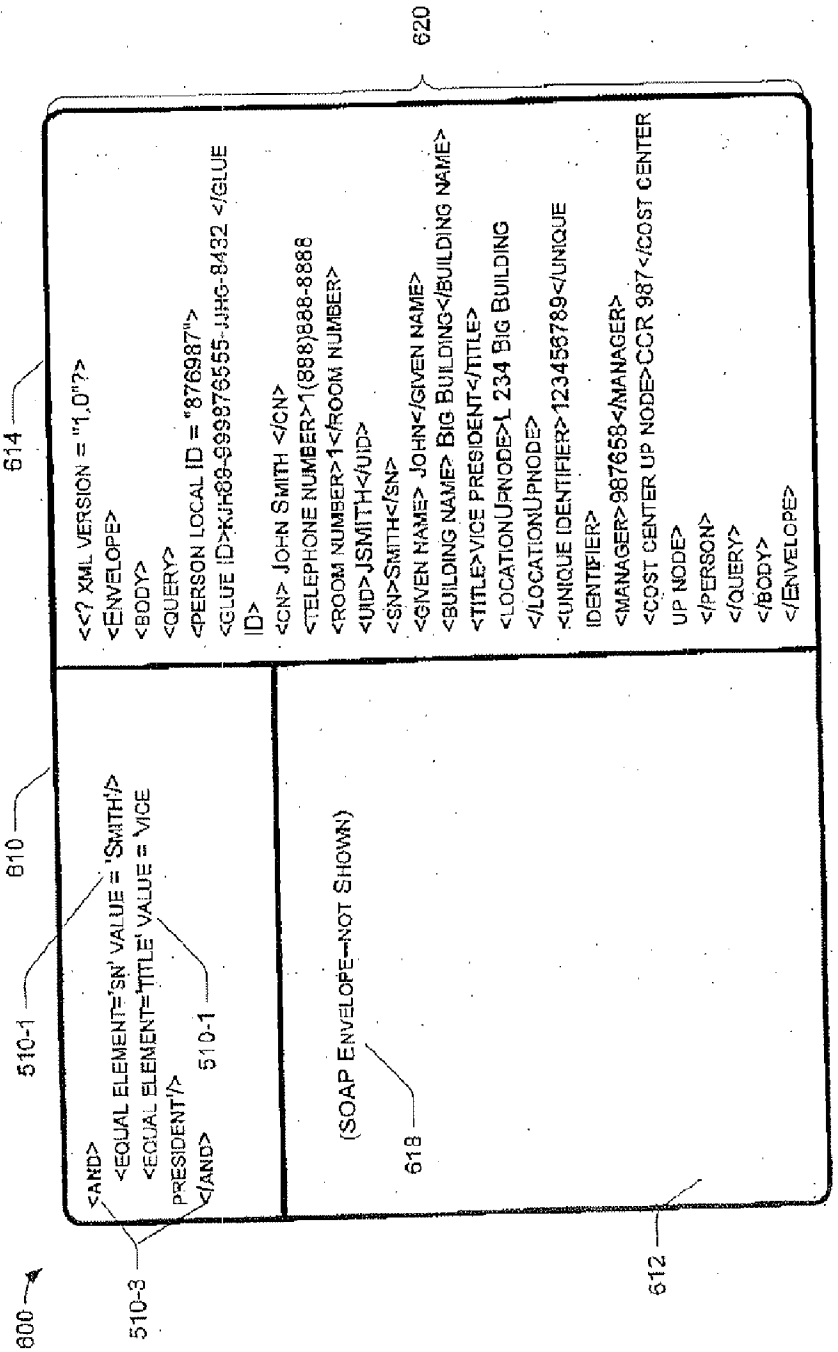

600
610
612
614
620
510-1
510-3
510-1
<AND>
<EQUAL ELEMENT='SN' VALUE = 'SMITH'/>
<EQUAL ELEMENT='TITLE' VALUE = VICE PRESIDENT'/>
</AND>
(SOAP ENVELOPE--NOT SHOWN)
618
<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY>
<PERSON LOCAL ID = "876987">
<GLUE ID>KJH89-999876555-JJH6-8432 </GLUE ID>
<CN> JOHN SMITH </CN>
<TELEPHONE NUMBER>1(888)888-8888
<ROOM NUMBER>1</ROOM NUMBER>
<UID>JSMITH</UID>
<SN>SMITH</SN>
<GIVEN NAME> JOHN</GIVEN NAME>
<BUILDING NAME> BIG BUILDING</BUILDING NAME>
<TITLE>VICE PRESIDENT</TITLE>
<LOCATIONUPNODE>L 234 BIG BUILDING
</LOCATIONUPNODE>
<UNIQUE IDENTIFIER>123456789</UNIQUE IDENTIFIER>
<MANAGER>987658</MANAGER>
<COST CENTER UP NODE>CCR 987</COST CENTER UP NODE>
</PERSON>
</QUERY>
</BODY>
</ENVELOPE>

Fig. 8

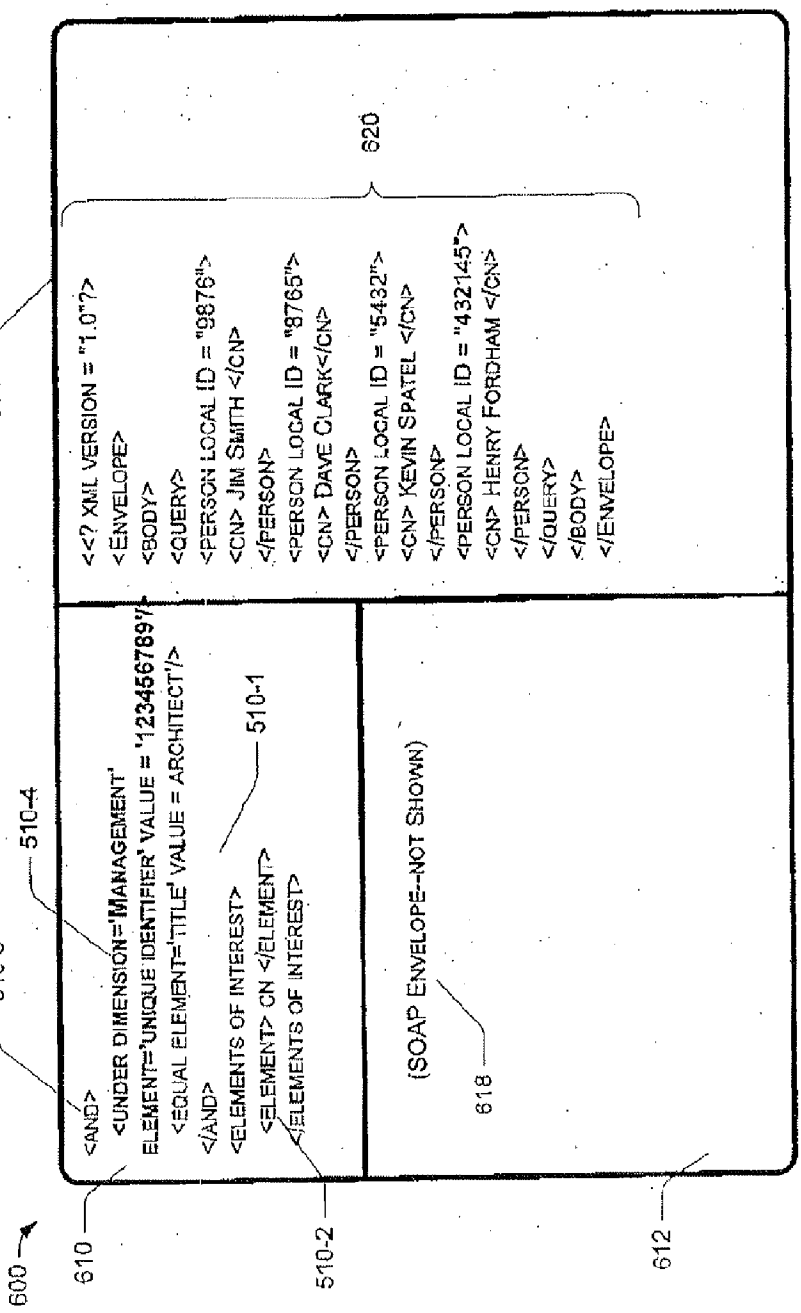

600
610
510-2
510-3
510-4
510-1
612
614
620
618
<AND>
<UNDER DIMENSION='MANAGEMENT'
ELEMENT='UNIQUE IDENTIFIER' VALUE = '123456789'/>
<EQUAL ELEMENT='TITLE' VALUE = ARCHITECT'/>
</AND>
<ELEMENTS OF INTEREST>
<ELEMENT> CN </ELEMENT>
</ELEMENTS OF INTEREST>
(SOAP ENVELOPE--NOT SHOWN)
<<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY>
<PERSON LOCAL ID = "9876">
<CN> JIM SMITH </CN>
</PERSON>
<PERSON LOCAL ID = "8765">
<CN> DAVE CLARK</CN>
</PERSON>
<PERSON LOCAL ID = "5432">
<CN> KEVIN SPATEL </CN>
</PERSON>
<PERSON LOCAL ID = "432145">
<CN> HENRY FORDHAM </CN>
</PERSON>
</QUERY>
</BODY>
</ENVELOPE>

Fig. 9

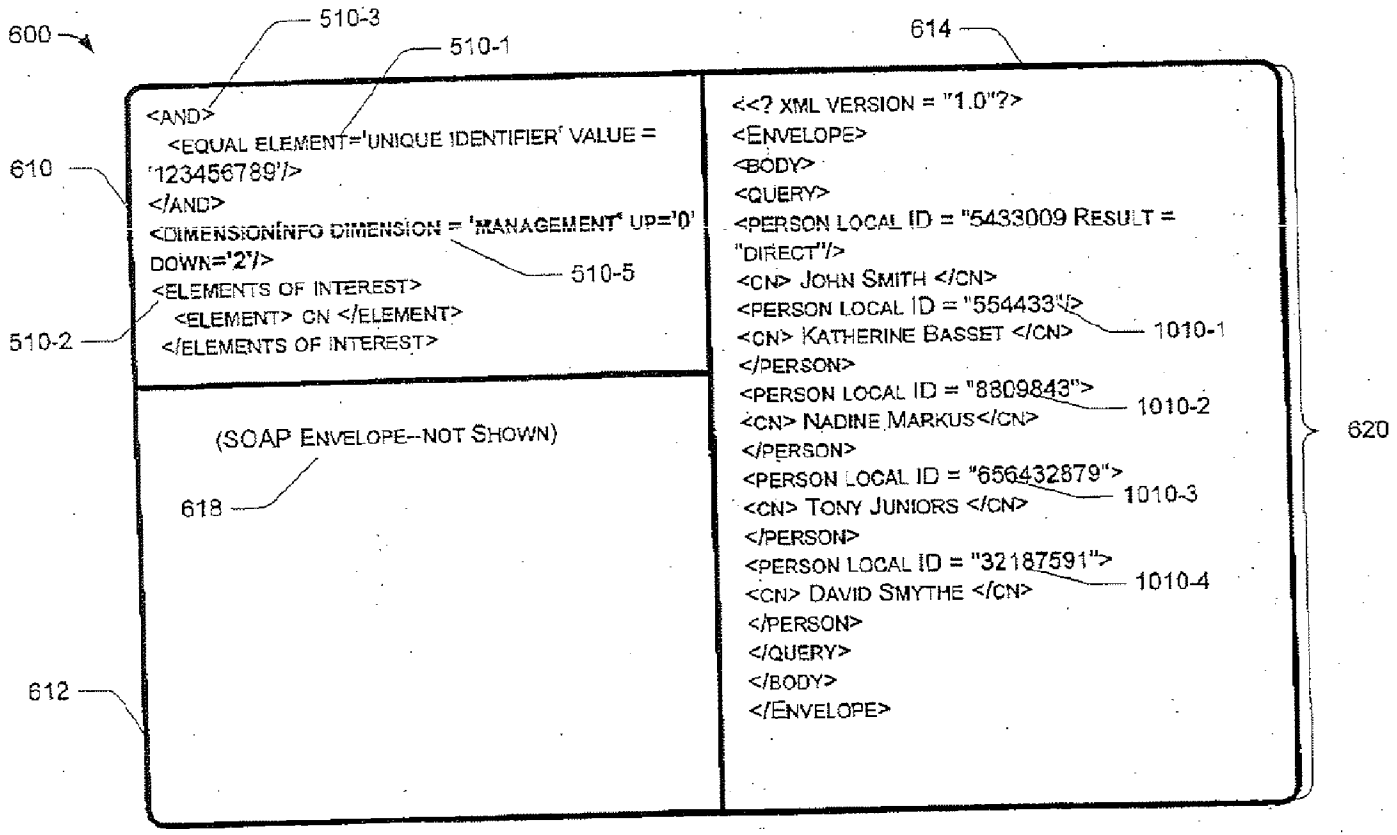


Fig. 10

(119) 特開2002-297433

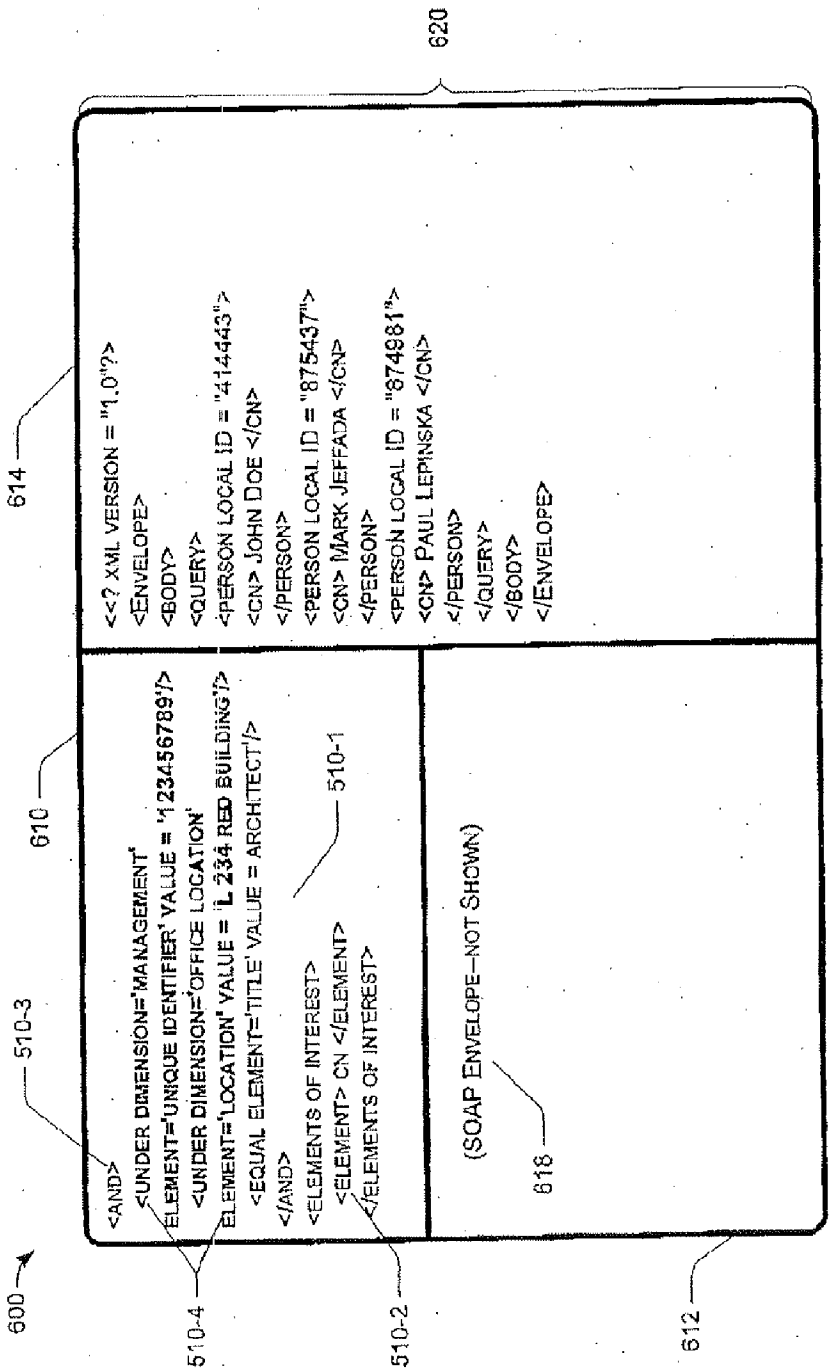

600
510-3
610
614
620
510-4
510-2
612
618
510-1
<AND>
<UNDER DIMENSION='MANAGEMENT'
ELEMENT='UNIQUE IDENTIFIER' VALUE = '1234567891'/>
<UNDER DIMENSION='OFFICE LOCATION'
ELEMENT='LOCATION' VALUE = 'L 234 RED BUILDING'/>
<EQUAL ELEMENT='TITLE' VALUE = ARCHITECT'/>
</AND>
<ELEMENTS OF INTEREST>
<ELEMENT> CN </ELEMENT>
</ELEMENTS OF INTEREST>
(SOAP ENVELOPE—NOT SHOWN)
<<? XML VERSION = "1.0"?>
<ENVELOPE>
<BODY>
<QUERY>
<PERSON LOCAL ID = "414443">
<CN> JOHN DOE </CN>
</PERSON>
<PERSON LOCAL ID = "875437">
<CN> MARK JEFFADA </CN>
</PERSON>
<PERSON LOCAL ID = "874981">
<CN> PAUL LEPINSKA </CN>
</PERSON>
</QUERY>
</BODY>
</ENVELOPE>

Fig. 11

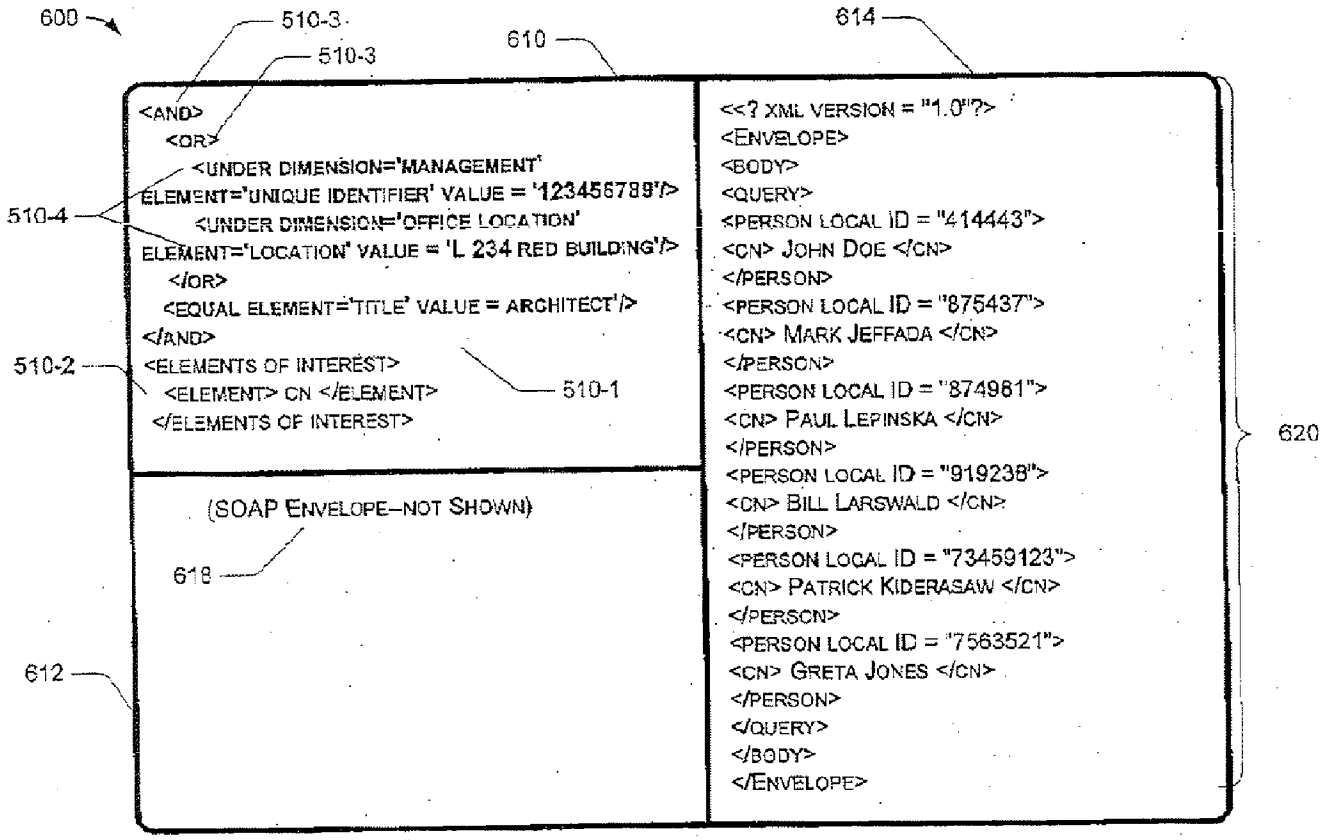


Fig. 12

(121)
特開2002-297133

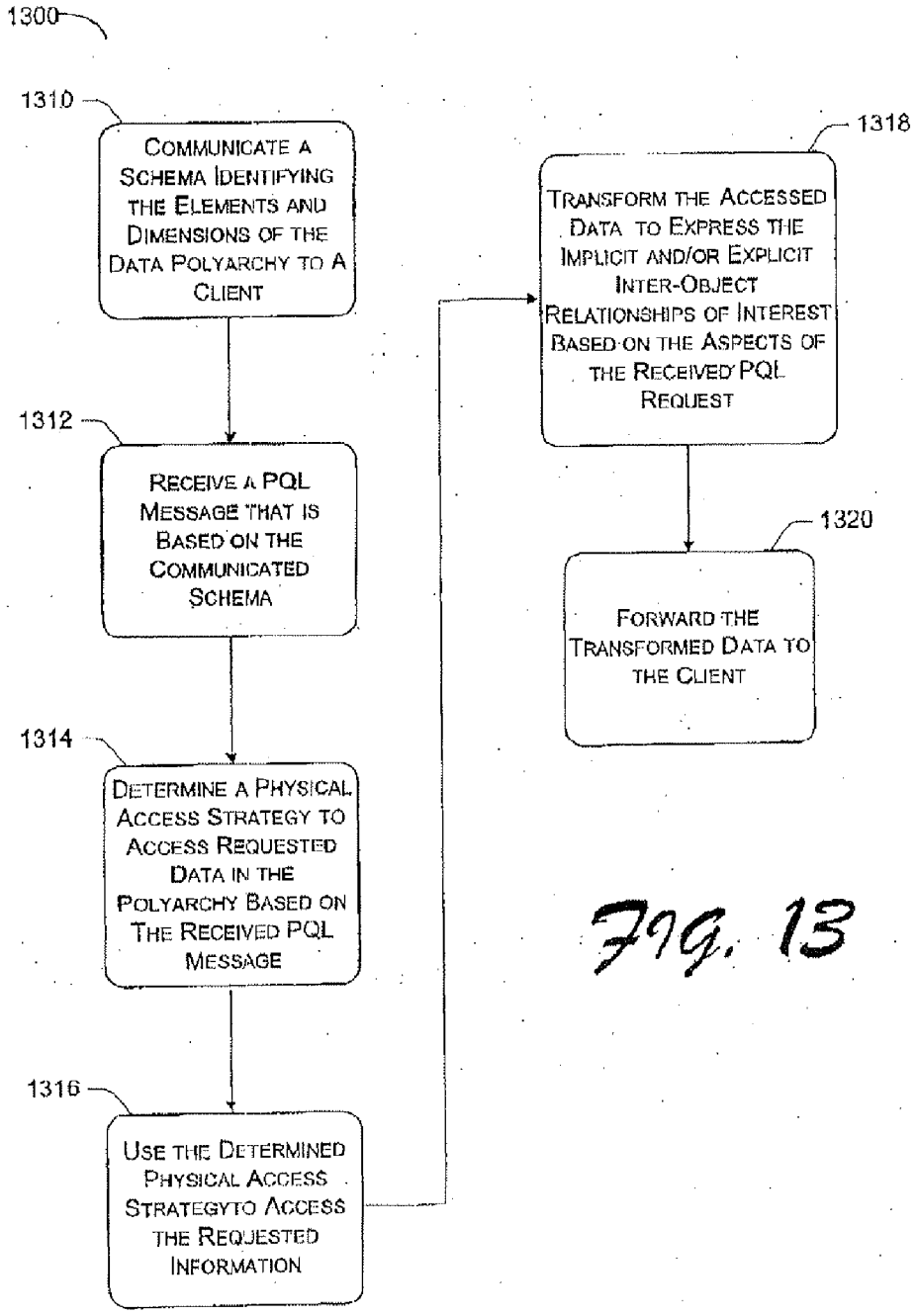

1300
1310
COMMUNICATE A SCHEMA IDENTIFYING THE ELEMENTS AND DIMENSIONS OF THE DATA POLYARCHY TO A CLIENT
1312
RECEIVE A PQL MESSAGE THAT IS BASED ON THE COMMUNICATED SCHEMA
1314
DETERMINE A PHYSICAL ACCESS STRATEGY TO ACCESS REQUESTED DATA IN THE POLYARCHY BASED ON THE RECEIVED PQL MESSAGE
1316
USE THE DETERMINED PHYSICAL ACCESS STRATEGYTO ACCESS THE REQUESTED INFORMATION
1318
TRANSFORM THE ACCESSED DATA TO EXPRESS THE IMPLICIT AND/OR EXPLICIT INTER-OBJECT RELATIONSHIPS OF INTEREST BASED ON THE ASPECTS OF THE RECEIVED PQL REQUEST
1320
FORWARD THE TRANSFORMED DATA TO THE CLIENT

FIG. 13

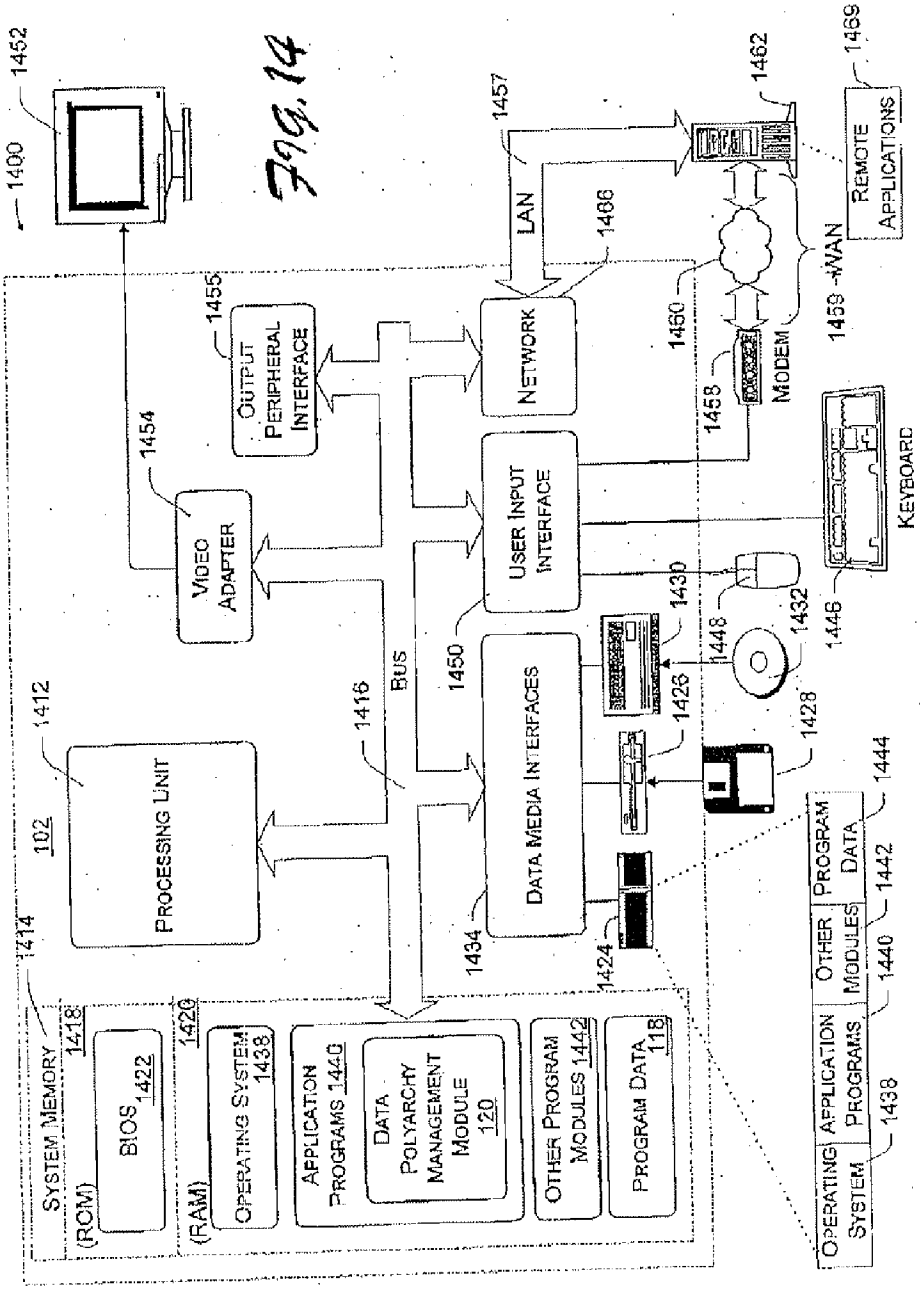


Fig. 14

1. Abstract

The described arrangements and procedures dynamically generate a data polyarchy from information received from a data store (e.g., a directory or database). The data polyarchy represents multiple hierarchies of inter-object relationships based on values of attributes of the objects. These multiple hierarchies are generated and represented in a manner that is independent of object naming and predetermined hierarchical data structures.

2 . Representative Drawing

FIG. 4

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 974 895 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
26.01.2000 Bulletin 2000/04

(51) Int. Cl.[7]: **G06F 9/44**, H04L 29/06

(21) Application number: **99107509.4**

(22) Date of filing: **14.04.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **03.07.1998 US 110748**

(71) Applicant:
MITSUBISHI DENKI KABUSHIKI KAISHA
Tokyo 100-8310 (JP)

(72) Inventor: **Peng, Luoscheng**
San Jose, California 95117 (US)

(74) Representative:
Pfenning, Meinig & Partner
Mozartstrasse 17
80336 München (DE)

(54) System for user control of version synchronization in mobile computing

(57) A universal system is provided for synchronizing servers which accommodates wide area mobile computing while at the same time making the process more efficient. The system includes a network of primary servers with high performance reliable links making up the backbone of the synchronization process to which secondary servers are linked via typically less reliable links. Moreover, synchronization from a mobile computer can be done whether in client/ server mode, or peer-to-peer to support any topology of secondary servers. In one embodiment while the primary servers are automatically and frequently synchronized, synchronization of the secondary servers is under the control of the user which prevents unintended synchronization. A summarizing version vector is used to minimize the amount of data transmitted by avoiding the necessity for exchanging version vectors for individual objects. This summarizing version vector also permits differential synchronization using summarizing version vectors and update stamps, the generation of a latest common version vector to purge off differential updates on a server, restart of synchronization from the point of previous failure with data from an unaffected server, and fine grain synchronization by permitting a differential update as the atom of data to be transmitted. Additionally, the system automatically switches between whole object synchronization and differential synchronization. Further, the subject system permits synchronization between different systems because the semantics of the data is segregated from the synchronization due to extracting updates in a standard format and synchronizing based on a standard protocol.

EP 0 974 895 A2